TECHNICAL REPORT ARSCD-TR-81007

MEDIAN FILTER NOISE IMPROVEMENT OF DIGITAL IMAGERY

GARY SIVAK

JUNE 1981

US ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND
FIRE CONTROL AND SMALL CALIBER
WEAPON SYSTEMS LABORATORY
DOVER, NEW JERSEY

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

Destroy this report when no longer needed. Do not return it to the originator.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TECHNICAL REPORT ARSCD-TR-81007	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* MEDIAN FILTER NOISE IMPROVEMENT OF DIGITAL IMAGERY		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* GARY SIVAK		8. CONTRACT OR GRANT NUMBER*(s)*
9. PERFORMING ORGANIZATION NAME AND ADDRESS ARRADCOM, FC&SCWSL Fire Control Div (DRDAR-SCF-IO) Dover, NJ 07801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project 12161101A011A
11. CONTROLLING OFFICE NAME AND ADDRESS ARRADCOM, TSD STINFO Div (DRDAR-TSS) Dover, NJ 07801		12. REPORT DATE June 1981 13. NUMBER OF PAGES 64
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
Image processing
Median filtering
Digital imagery
Signal-to-noise ratio
Automatic target recognition system
Adaptive window median filtering (AWMF)

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
Designs for fire control automatic target recognition systems require poise-free or low-noise digital imagery of visible and infrared targets. Median filter noise cleaning of digital imagery was investigated to determine the extent to which different size median filter windows will improve the signal-to-noise ratio of a digital image with the least deterioration or degradation to the shape or position of the edges of a target. Examples of median filtering and adaptive window filtering (AWMF) are presented before and after
(CONTINUED)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (Cont)
noise cleaning, and an improvement over AWMF is proposed to remedy its display deficiencies. Recommendations are offered on how to proceed to obtain more meaningful results and how to extend the simulation to two dimensions before performing test on actual digital imagery data.

CONTENTS

	Page
Introduction	1
Median Filtering	1
Operation	3
Innovations	
Analysis	11
Strategy	11
Program SIMU1D	15
Development	15
Subroutine READIN	16
Subroutine CALC	17
Subroutine IGTAB	17
Subroutine SIMP	18
Subroutine STORIT	18
Subroutine LINTERP	19
Subroutine SETEJ	19
Subroutine FILLEJ	19
Subroutine MD1DW	20
Subroutine DSORT	20
Subroutine FINDEJ	20
Subroutine MSE	21
Subroutine DOMORE	21
Display Subroutines	21
Program LOCKON	22
Programs EJSHOW and PULPLT	22
Program EJSHOW	22
Program PULPLT	23
Preliminary Results	23
Recommendations	27
References	29
Bibliography	30
Appendixes	
A. Program SIMU1D	33
B. Program LOCKON	43
C. Program EJSHOW	51
D. Program PULPLT	61
Distribution List	65

FIGURES

		Page
1	Median filter	2
2	Example of median filtering	4
3	Noisy square wave, 10% Gaussian noise	5
4	Noisy square wave, 10% Gaussian noise, filtering window width 3	6
5	Adaptive window median filtering weighting functions	9
6	One-dimensional analytic procedure	12
7	One-dimensional edge profile	13
8	Filtered versus input Gaussian noise: one-dimensional simulation	24
9	One-dimensional noisy edge	25
10	One-dimensional edge analysis, 10% Gaussian noise, filtering window width 3	26

INTRODUCTION

Whether a battle scene is presented on a display for human interpretation or is processed by an automatic target recognition system, a need exists in fire control for noise-free or, at least, low-noise digital imagery of visible and infrared targets of interest. In this report, initiation and implementation of an innovative effort are presented to investigate the median filter noise cleaning of digital imagery. Specifically, the goal is to determine the extent to which different size median filter windows will improve the signal-to-noise ratio of a digital image with the least possible amount of deterioration or degradation to the shape or position of the edges of a target of interest. Signal-to-noise ratio improvement is a relevant figure of merit to use for measuring the performance and usefulness of a median filter because for two scenes with similar kinds of noise characteristics (zero-mean Gaussian, for example), the image with the higher of the two signal-to-noise ratio has the better quality.

In this report, median filtering and adaptive window median filtering (AWMF) are first explained and illustrated with examples of a pictorial scene and digital signals before and after noise cleaning. Next, an improvement over AWMF is proposed to remedy its display deficiencies. Then, an analytical strategy is developed by which the median filter reduction can be measured through representation of one-dimensional edge profiles by the integral of a Gaussian distribution. A one-dimensional computer simulation, the program/modular subroutine package program SIMU1D is developed to implement the analysis and plot post-processing versus input signal-to-noise levels. The implementation of programs LOCKON and EJSHOW, used to test the validity of the preceding analysis, are explained. After a discussion of the preliminary results obtained, recommendations are presented on how to continue the work to obtain more meaningful results and how to extend the simulation to two dimensions before tests are performed on actual digital imagery data.

MEDIAN FILTERING

Operation

The nonlinear signal-smoothing technique of median filtering was first developed by John Tukey and applied in the mid-1970's to the processing of digital speech signals (ref 1). When applied to two-dimensional image scenes, the technique has the capability to reduce the prominence of troublesome random noise spikes and regions in the picture.

Figure 1 shows how two-dimensional median filtering is performed. Take a window box M picture elements (pixels) high and N pixels wide; and place it in the upper left corner of the image. M and N are odd numbers. In this example, M is 3 and N is 5, although in practice the use of a square window is convenient as long as the dimensions are odd integers.

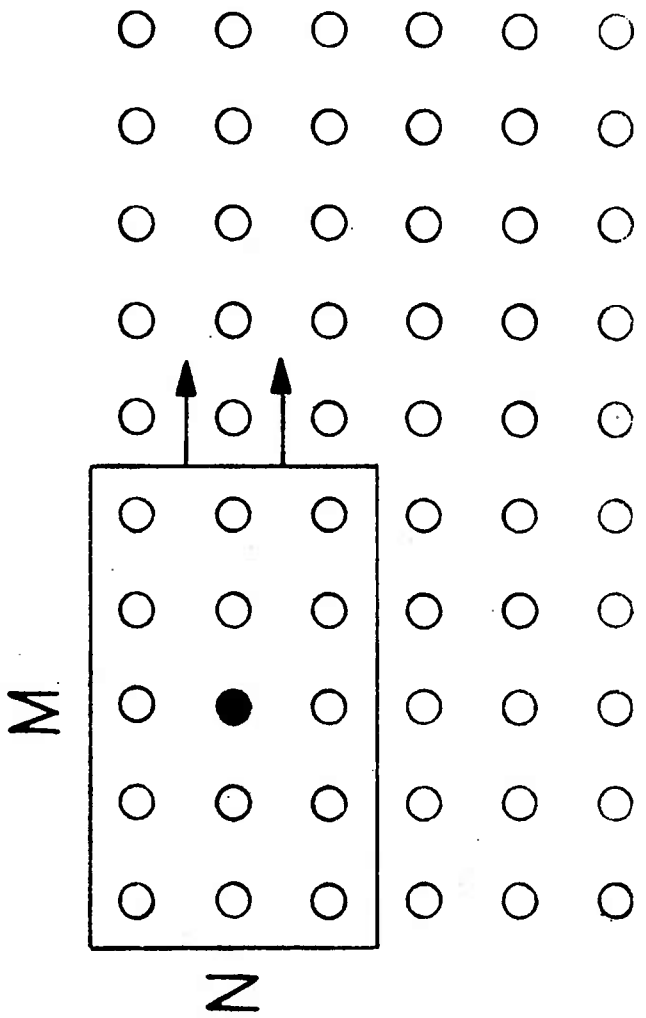

Figure 1. Median filter

To implement the algorithm, sort the values contained by the window box into consecutive order and replace the pixel in the center of the box by the median value, that is, the middle entry in the sorted list. The algorithm is continued by sliding the window box one pixel to the right and performing the sort-and-center pixel replacement again. The box is then allowed to slide across each line of pixels in the image and slide down to the bottom of the scene.

Figure 2 shows an example of median filtering of an actual scene. The picture of the USC cheer leader is shown before and after being operated on by a one-dimensional median filter window of width 5 (ref 2).

Figure 3 shows a computer graphics plot of a noisy square-wave pulse with a 10 percent concentration of zero-mean Gaussian random noise. Figure 4 shows the square-wave pulse after it has been operated on by a one-dimensional median filter window of width 3 units. As the signal is improved, note that the median filter has the important property to preserve the edges of the square-wave pulse. Since the proper Fourier representation of a square-wave pulse contains an infinite number of terms decreasing only slowly by 1/N, where N is an odd integer multiple of the fundamental frequency, a large number of terms or frequency components is required to accurately represent the square wave with good convergence and small error. Therefore, an ordinary band pass filter cannot do as good a job in removing the random noise from the signal without destroying its shape, phase, or other characteristics. The only quantities that two-dimensional median filters will destroy are the vertices or single -corner pixels of sharp-cornered objects.

Innovations

An extension of the median filtering algorithm, an improved filtering technique currently in use, is the procedure known as adaptive window median filtering (AWMF). In this process, the filtering window dimensions are allowed to vary inversely as the amount of edge or gradient in different regions of the image increases. The reason for this is that a big median filter window will attenuate noise by a greater factor than a smaller one, but at the expense of smoothing over edge details less than one-half the window width. For example, in regions of the picture where the gradient information or edge content is a maximum, one can use a window box of width 1, i.e., no filtering. In regions of the picture where the edge content or gradient is in the midrange between the minimum and maximum, one may filter the picture area with a 3-by-3 window matrix, for example, thus eliminating noise spike signal transmission errors.

Finally, in areas of the scene where the edge content is at a low, the pixel information simply consists of texture background and noise. The 3-by-3, two-dimensional, median filtering window can be expanded to a 5-by-5 matrix and will remove 2-pixel noise clusters. That is, in relatively quiet areas of the scene, from the standpoint of image

A. Before.

B. After.

Figure 2. Example of median filtering.

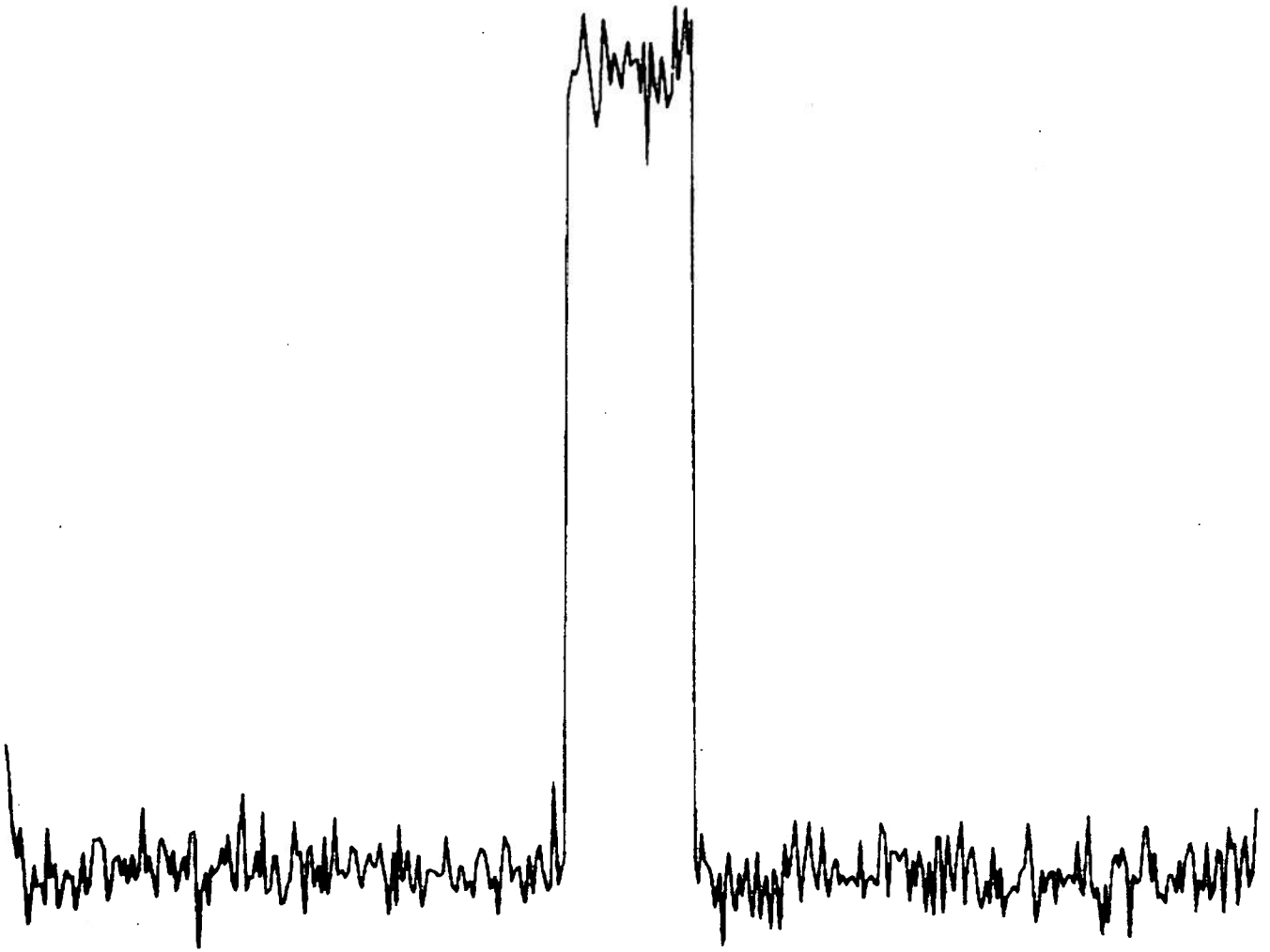

Figure 3. Noisy square wave, 10% Gaussian noise

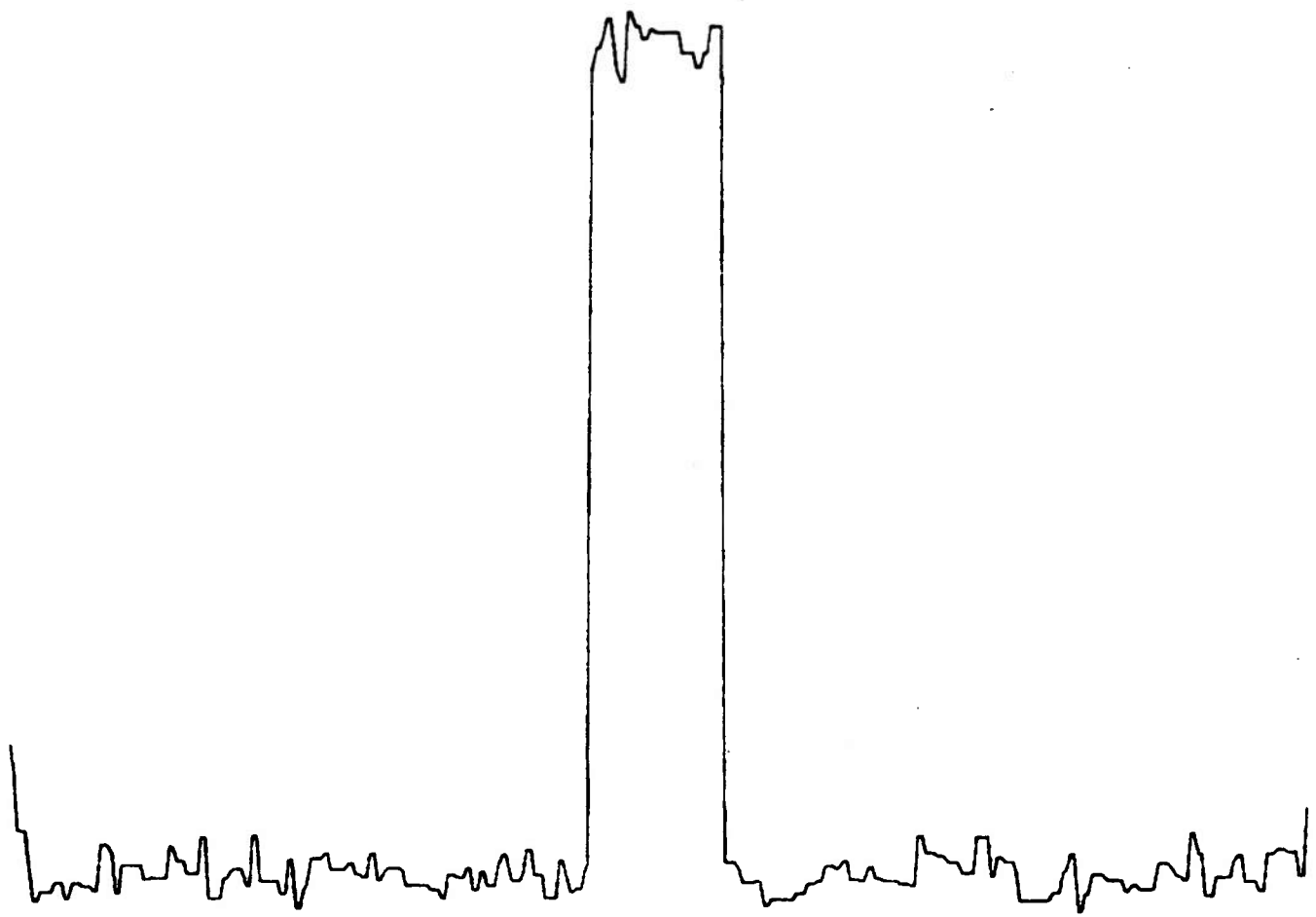

Figure 4. Noisy square wave, 10% Gaussian noise, filtering window width 3

information of concern, where little target information is present, one simply lets the two-dimensional median filter window size expand to a maximum to be able to filter the largest noise-point cluster that can possibly be done.

The values of the gradient are read off a gradient map, each point of which is computed for each point of the original picture for a pixel value P_{ij} where i represents the ith row of the image and j represents the jth column. The magnitude of the gradient G_{ij} is computed from the horizontal (right minus left) and verticle (upper minus lower) pixel differences surrounding the point as follows:

$$G_{i,j} = \sqrt{(P_{i,j+1} - P_{i,j-1})^2 + (P_{i+1,j} - P_{i-1,j})^2} \qquad (1)$$

For extremely large picture areas, 512 by 512 pixels, for example, to save computer time, one may in practice use:

$$G_{ij} = \left| P_{i+1,j} - P_{i-1,j} \right| + \left| P_{i,j+1} - P_{i,j-2} \right| \qquad (2)$$

For very noisy images, a more stable gradient can be obtained by spreading the calculation over a larger pixel area to smooth out random pixel fluctuations and variations that attenuate as $\frac{1}{n}$ for N number of pixels. For example, in the 3-pixel by 3-pixel box surrounding a given pixel, i.e., the cluster comprising a given pixel and its eight nearest neighbors:

$$G_{ij} = \left| \text{SUM OF TOP THREE PIXELS MINUS SUM OF BOTTOM THREE PIXELS} \right| + \left| \text{SUM OF RIGHT THREE PIXELS MINUS SUM OF LEFT THREE PIXELS} \right| \qquad (3)$$

Performing adaptive window median filtering however is difficult, because as the window expands, one will re-enter portions of the scene that have already been processed. Alternatively, to prevent this, one can let the center of the median filter window box skip nonuniformly across the image. This, however, prevents the center of the window box from contacting some points and, if they are of a spurious noise, they will not be replaced upon sorting. Also, another serious problem is that changing the window size produces uneven noise attenuation, more so for the larger windows. The results thereby obtained can be rather uneven, a very sloppily filtered scene with such artifacts as regions of blocking and false detail.

To remedy some of these display difficulties, an improvement over the technique of adaptive window median filtering is proposed by which the

output results of filtering the image by several two-dimensional windows are combined, being weighted by factors that are continuous functions of the gradient (ref 3). In this filtering approach, as the magnitude of the gradient varies throughout the image, the output picture does not undergo sudden transitions in the gray level since the median filtering windows switch from large to medium to small. For example, let P_3 be a point in a picture filtered by a 3-by-3 window, let P_5 be the result of filtering by a 5-window, and P_7 be the result of filtering by a 7-window. An output pixel P is then computed as the weighted sum:

$$P = A \times P_3 + B \times P_5 + C \times P_7 \qquad (4)$$

where (as stated before) A, B, and C are continuous functions of the gradient (figure 5). To ensure that the pixel values are properly displayed over the dynamic range of an output display, i.e., to ensure that gray levels are not washed out or contrast lost, 1 must also require:

$$A + B + C = 1 \qquad (5)$$

As the value on the gradient map of the digitized image in question rises from a minimum to a maximum, the function A rises from a minimum of 0 to a maximum of 1. Similarly, as the gradient goes from a minimum to a maximum, the function B rises from a minimum to a maximum at the gradient midrange values and then falls to a minimum value as the image gradient attains a maximum. Finally, as the image gradient goes from a minimum to a maximum value, the function C falls from a maximum of 1 to a minimum of 0.

If these criteria for describing the continuous functions of the gradient, A, B, and C, are adhered to, what then occurs is as follows: In regions of the image where the gradient is small, A and B are approximately equal to 0, C is approximately equal to 1, and the filtered output of the largest window has the predominant weight. In regions of the image where the values of the gradient are in the midrange, the function B is the maximum and the midrange window-size output predominates. Where the gradient is near a maximum in the scene, A is approximately equal to 1, and B and C are approximately 0. Here, the filtered image is dominated by the output of the smallest median filter window, as is desired. The problem of sudden window-size transitions simply no longer exists.

The author now proposes a working scheme by which various choices of the continuous functions A, B, and C of image gradient can be generated and applied to the filtering of digitized frames of imagery of interest. Allow the following conditions to exist:

$$Y = \sqrt{A} \qquad X = \sqrt{B} \qquad Z = \sqrt{C} \qquad (6)$$

$$P = A \cdot P_3 + B \cdot P_5 + C \cdot P_7$$

$$A + B + C = 1$$

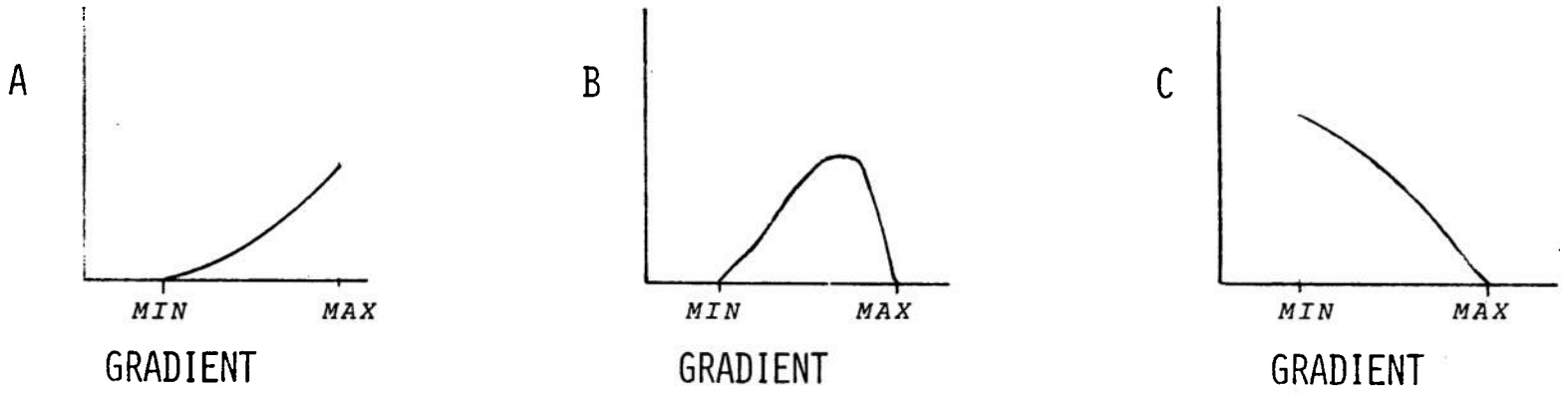

Figure 5. Adaptive window median filtering weighting functions

Then, from equation 5 one has:

$$X^2 + Y^2 + Z^2 = 1 \tag{7}$$

$$X = \cos(\phi)\sin(\theta) \qquad Y = \sin(\phi)\sin(\theta) \qquad Z = \cos(\theta) \tag{8}$$

Equation 8 represents the position of a point on the surface of a sphere as parametrized by two angles: θ measured down from the sphere's north pole and ϕ measured counterclockwise from some reference meridian that cuts through the sphere's equator.

To generate the functions A, B, and C as the image gradient varies from a minimum to a maximum value, one can, for example, simply force θ and ϕ to vary from zero to 90 degrees, thus tracing out a curving path on the surface of the sphere, from its north pole to its equator. Specifically, if the angles θ and ϕ are set equal to each other and to a parameter α, then from equations 6 and 8:

$$A = \sin^4(\alpha) \qquad B = \cos^2(\alpha)\sin^2(\alpha) \qquad C = \cos^2(\alpha) \tag{9}$$

When the gradient varies from a minimum to a maximum, the parameter α can be forced to vary from zero to 90 (in degrees), if one has:

$$\alpha = 90 \left(\frac{G - G_{MIN}}{G_{MAX} - G_{MIN}} \right) \tag{10}$$

where G, G_{MIN} and G_{MAX} are each the given minimum and maximum values of the image gradient, respectively.

In this example, the path described on the surface of the unit sphere is a curve, sweeping down from the north pole of the surface of the sphere in a southeasterly direction to a point down on the equator at 90^o east longitude. Note that each possible path that one can draw on the surface of the sphere represents a particular combination of median filter window outputs from which to form a unique resultant picture. An infinite number of possible paths exists on the surface of the sphere to choose from to produce the best output picture as a weighted sum of input images operated on by combinations of different two-dimensional median filter windows. For combinations of four or more different input median filter windows per output picture, this analysis can conceivably be extended to uniquely link window combinations with curves drawn on the surface of spheres in four or higher-dimensional hyperspaces.

ANALYSIS

Strategy

Besides reducing the signal-to-noise ratio in an image, one hopes, but cannot be certain, that a particular median filter will not affect the shape or structure of the contour of a target or the connectivity of its component parts. A good way to begin to understand how a median filter may affect a target is to study its effects on the edge profiles that surround a given target and separate it from the bordering background information in the total scene.

No edge in digital imagery is perfectly sharp, but is characterized by a sloping rise from low to high, i.e., from a cool or dim background to a hot or bright target. This monatonically increasing ramp can occupy many or as few as two pixels, a low gray level and a high gray level. Part of the problem with studying median filters is investigating how the slope or width of these edge profiles may be affected by the operation of the filter. Also, the simplest case to start with involves the effect of a one-dimensional median filter window of width N, where N is an odd number, on a ramp which represents a one-dimensional slice through the edge profile of a target in an image under investigation.

In terms of one-dimensional digital signal processing, the edge ramp E can be considered as a digital function E(t) where t is a measure of the pixel location in the edge. For a given working field width of 101 pixels, t is only defined for the integral values 0, 1, 2, ... through 100.

A flow diagram depicting the strategy of study of the one-dimensional case is shown in figure 6 as follows: Take an input edge pure signal E(t) and add to it a given percentage intensity P_n of zero-mean Gaussian random noise $N_1(t)$ to get a noisy edge profile F(t). (Gaussian noise is characteristic of the output of mechanisms responsible for various sensor and transmission noise sources and is, therefore, a good model of noise to use in this study.) Then operate upon F(t) with a given one-dimensional median filter window to obtain an unknown filtered noisy edge profile G(t). Assuming that the slope of the edge profile G(t) may have been distorted by the action of the median filter, put G(t) through an edge estimator to find the best fit pure signal H(t). If one defines:

$$N_2(t) = G(t) - H(t) \tag{11}$$

then $N_2(t)$ is a measure of the postprocessing noise.

The input edge profile ramp can be of any desired width and defined to lie within a 101-pixel working field as shown in figure 7. The particular ramp shown in the figure rises from a low of zero to a high

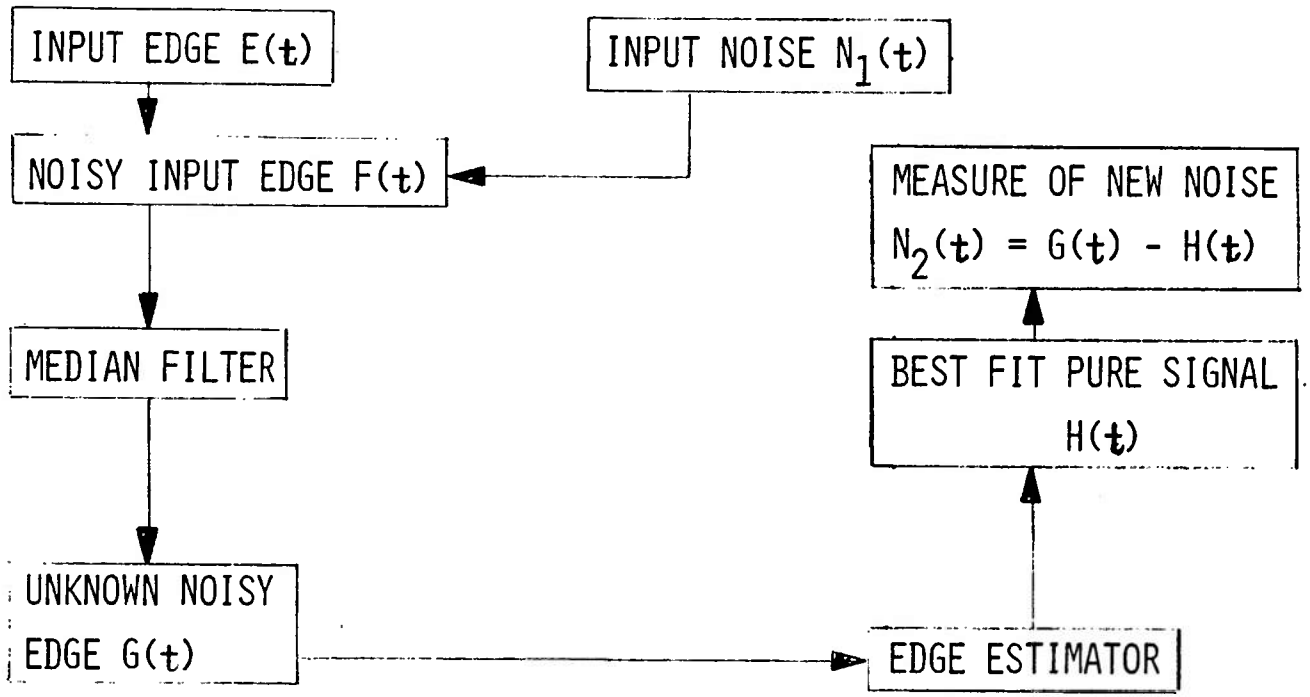

Figure 6. One-dimensional analytic procedure

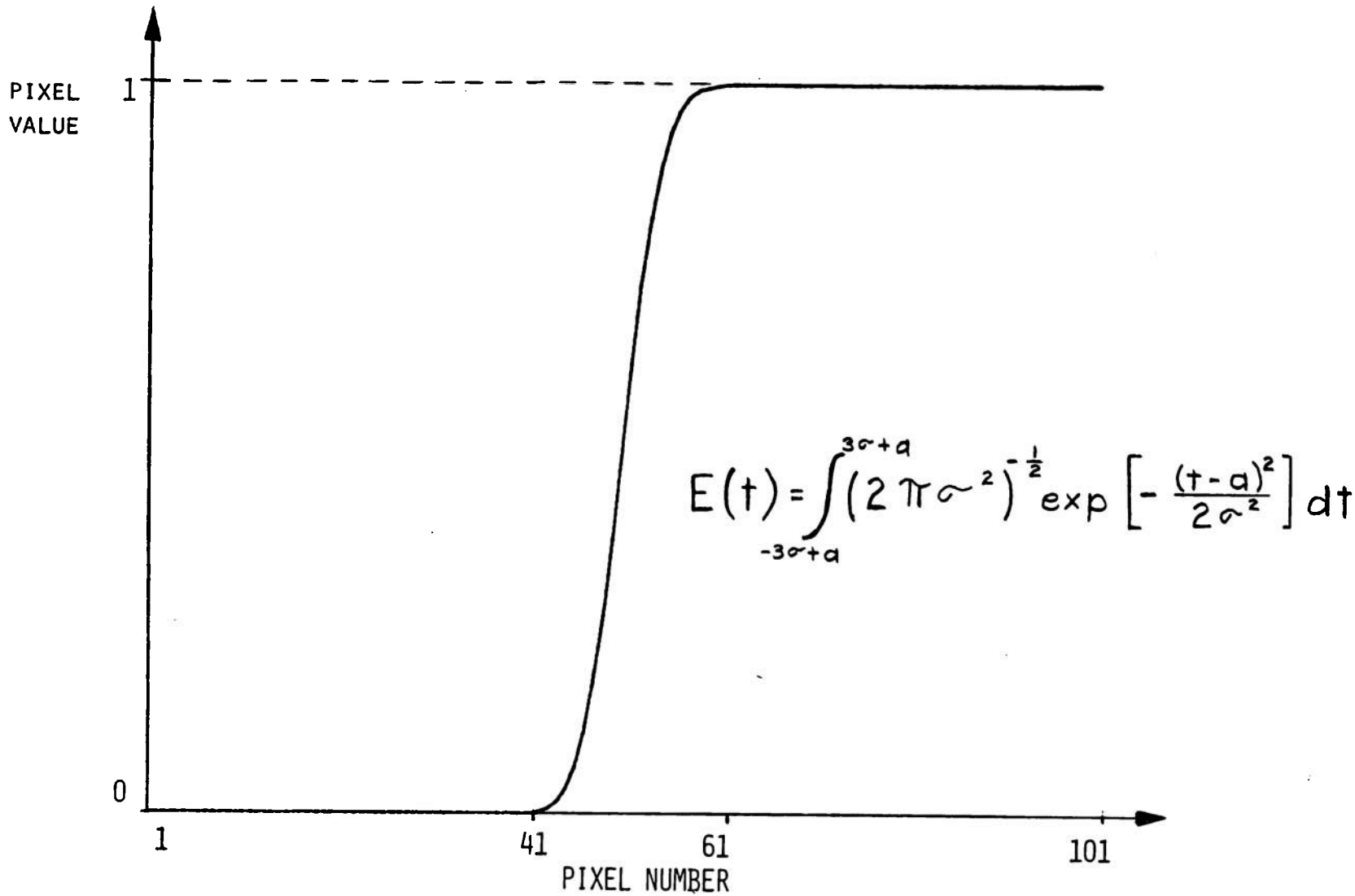

Figure 7. One-dimensional edge profile

value of one, from pixels 41 through 61 in the field of 101. The shape of the ramp is characterized by the integral of a Gaussian distribution:

$$F(t) = \int_{-3\sigma+A}^{3\sigma+A} \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) \left(\text{EXP}\left[-\frac{(t-a)^2}{2\sigma^2}\right]\right) dt \qquad (12)$$

This model of a smooth edge profile has no corners or discontinuities and is characteristic of the way edges are represented on photographic film, the source of some of the author's test data imagery of infrared targets. Therefore, the representation of an edge profile by the integral of a Gaussian distribution is a good model to use for purposes of investigation and is the one that will be considered from now on.

At this point, for a given percentage intensity of Gaussian noise P_n , the magnitude of the edge ramp is scaled so that no pixel is likely to exceed the maximum value Gramax (Fortran variable name). For the signal and noise contributions V_s and V_n, respectively, to the edge profile ramp F(t), one has:

$$V_n = \left(\frac{P_n}{100}\right)\left(\frac{\text{GRAMAX}}{\text{SIGMAS}}\right) \qquad V_s = 1 - \frac{P_n}{100}\ \text{GRAMAX}$$

$$F(t) = (V_s)\ (\text{RAMP}) + (V_n)\ (\text{GAUSIAN RANDOM NUMBER}) \qquad (13)$$

Here, SIGMAS is set to a value of three standard deviations. For example, if P_n is 10%, the 101 noise point contributions (Gaussian random number times V_n) will most likely not exceed 0.1 times GRAMAX. The reason for this is that a 0-mean Gaussian random number distribution with standard deviation equal to 1 (which can be approximated, for example, by the random number generator output of a computer) will exceed 3 only about 1 time out of 370. The signal contribution (V_s times a ramp between 0 and 1) will definitely not exceed 0.9 times GRAMAX so that adding on the noise contribution scales the pixels up to a maximum value of GRAMAX.

After the given median filter operates on F(t), the unknown noisy filtered edge G(t), which has been produced, may be distorted. One must therefore find the new pure signal, the best fit H(t), to which this unknown edge best corresponds.

First, assume that the noise $N_1(t)$ is always zero, i.e., the median filter operates solely upon the input edge E(t) as follows:

$$M(E(t)) = E(t) + \Delta_1(t) \tag{14}$$

where M is the nonlinear median filter operator and where $\Delta_1(t)$ is the amount of degradation added to the input signal. In other words, $\Delta_1(t)$ is simply the degradation that, when added to the input signal, will produce any distortions for which the median filter may be responsible. If the median filter operator M is now applied to the noisy edge ramp F(t), one obtains:

$$\begin{aligned} M(F(t)) &= M(E(t)) + N_1(t)) \\ &= F(t) + \Delta_2(t) \end{aligned} \tag{15}$$

For the median filter, the question that arises is, if equation 14 is true and if the operation of the median filter M on E(t) produces a degradation $\Delta_1(t)$, what can be said about the degradation $\Delta_2(t)$ in terms of E(t), $N_1(t)$, and $\Delta_1(t)$ as produced by the operation of the median filter on the sum of E(t) and $N_1(t)$? The problem is that for linear operations a great deal can be said about this question; but, for nonlinear procedures, such as median filter algorithms, nothing in general can be stated in terms of the results produced by perturbing a given input system by a given function of time. Therefore, numerical techniques by computer simulation become necessary if one hopes to extract more useful information about this problem.

To try to determine the new signal H(t) to which this unknown filtered noisy edge G(t) best corresponds, let the original ramp endpoints vary in and out by five units over eleven positions each. As the left ramp endpoint varies between the values 36 through 46 and the right one from 56 through 66, a set of 121 clean ramp signals will be generated. Scale each one without noise by the factor GRAMAX, as in equation 13, compare it with G(t), and generate the mean square error by adding the squares of the differences of corresponding data points. The particular ramp with the lowest mean square error or noise $N_2(t)$ is the best fit to the filtered noise-cleaned edge G(t). The left and right endpoints of this signal ramp H(t) define the location and slope of the best fit edge ramp desired.

PROGRAM SIMU1D

Development

Program SIMU1D is a one-dimensional median filter study simulation, a comprehensive, interactive Fortran program/modular subroutine package

designed to implement the one-dimensional edge profile analysis (both discussed in the previous section and described in figure 6). For an edge ramp centered in the 101-pixel wide working area, filtered by a one-dimensional median filter of desired width, and for any desired number of trial runs per data point, the simulation calculates the percent intensity of Gaussian noise after median filtering versus the percentage of Gaussian noise in the ramp before the application of the filter. The short 20-line main control program first calls upon a subroutine to interactively read in all user-specified parameters such as edge and window widths and minimum and maximum input noise intensity percentages. Subroutines are then executed to process all the required calculations, write the results to an external file for storage, if desired, and to determine the minimum and maximum values of the output noise percentage data. Subroutines are then called upon to print or plot the output versus input data as specified by the user.

The following subsections are a summary description in order of both the coding of the various modular subroutines in the package and the individual tasks which they perform.

Subroutine READIN

This subroutine is the port through which the simulation interacts with the user and conversationally asks for and absorbs all required specification parameters for a given test run.

The subroutine first asks the user for the type of display desired. Either a tabular printout of the results or a graphical plot or both can be produced. The user next specifies the width of the centered edge to be analyzed, the window width of the one-dimensional median filter to be used, and the number of runs per data point. For example, if the number of runs is specified as 25, the edge of desired width has a given percentage intensity of Gaussian noise added to it, is filtered, and the best edge found with these three operations being repeated in 25 separate trials. The 25 output mean square error noise values are then averaged to reduce the effect of fluctuations. This average value is the desired output noise percentage intensity result.

The user then specifies the range of the input noise data, the minimum and the maximum percentages of input noise intensity, and the number of points to be calculated within this range. For example, specifying 10%, 20% and 11 points to the computer would cause the input noise intensity specifications to increments from 10% to 20%, a percentage point at a time, as 10%, 11%, 12%, ... 20%, for a total of 11 data values.

Finally, if a graphical plot is requested, the user is asked to specify the left, right, upper, and lower margin portions to surround the plot. For example, if 0.1 is specified in each case, 10% of the available plotting area bordering each of the four sides of the total field available for graphical plot will be left blank and unused. The graph will then

occupy the center 80% of the entire plotting field, leaving room for writing captions and labels, etc., on a hard copy of the graphical results.

Subroutine CALC

Having been given the edge and window sizes and the range on the input noise intensity percentages, this subroutine controls all calculations necessary to obtain the output noise intensity percentages to be displayed.

First, subroutine IGTAB is asked to store the values of the integral of a Gaussian, as defined in equation 12, in the 1001-point lookup table, Fortran array GI, from which the edge ramps are formed by interpolation. Next, as defined by the specified width of the desired edge and the amount of variation of its endpoints in and out, subroutine STORIT is used to load into the array ED all edge profiles that are to be used to construct any subsequent edge ramps. Then, for each input noise percentage intensity, the appropriate edge profile is read from the array ED into the 101-pixel length working field, Fortran array EJ. The profile is then moved from the left and centered in the array EJ by subroutine SETEJ. Then subroutine FILLEJ is used to scale the profile, as described in equation 13, to the maximum value GRAMAX, which is set to 255, the biggest number attainable by typical eight-bit image processing displays.

At this point, array EJ contains the noisy edge ramp F(t) of figure 6. A one-dimensional median filter of user-specified window width is applied by subroutine MD1DW. Subroutine FINDEJ is then applied to the resulting unknown edge profile G(t) to find the Gaussian ramp that best fits this profile with the lowest mean square error. The left and right coordinates of the ramp are contained in variables NL2 and NR2, respectively, and the mean square error is contained in the variable ERVAL.

If additional runs for the same data point are needed for subsequent averaging, they are done by subroutine DOMORE. The resulting mean square errors are then averaged within subroutine CALC. Note that, in equation 13, the input noise was scaled by the factor GRAMAX divided by SIGMAS. For proper error scaling, this must be removed; the mean square error is thus multiplied by the reciprocal factor SIGMAS divided by GRAMAX and then by 100 to obtain the output percentage of noise intensity. This entire procedure is then repeated until all output values of percentage noise intensity have been calculated.

Subroutine IGTAB

This subroutine calculates and stores in the lookup table GI 1001 values of the integral of a Gaussian or normal distribution from three standard deviations before the origin to three deviations after the mean at the origin. The values will then be used subsequently by subroutine STORIT to construct the required edge profiles. Specifically,

$$I(X) = \int_{-3}^{X} \frac{1}{\sqrt{2\pi}} \exp\left[\frac{-z^2}{2}\right] dz \qquad (16)$$

where x represents the number of standard deviations and varies uniformly from -3 to 3. If x ran from -00 to 00, the ramp would rise from 0 to exactly 1. However, most of the activity of the function is confined to within a few standard deviations about the origin. For practical purposes, 3.0 was chosen as sufficiently accurate for the program, the maximum value of the integral obtained by the ramp in equation 16 being 0.9973. For proper accuracy, one would desire that, when this maximum valve is scaled, that is, multiplied by GRAMAX = 255, it is correct to within half a pixel or at least 254.5. However, it turns out to be just a little bit short and yields 254.311. For correctness to the nearest pixel, the user can feel free to extend the limits on the integral in equation 16 and subroutine IGTAB from -3.0 and 3.0 to -3.1 and 3.1. This yields a maximum tabulated value of 0.99806 for a scaled value of 254.5065, correct to the nearest pixel.

Subroutine SIMP

This routine integrates a function partitioned into N steps between X_{MIN} and X_{MAX} by Simpson's rule. It is used by subroutine IGTAB to recursively generate new values of the accumulating integral of the Gaussian from ones previously known. The first value would be identically 0, being the integral from -3 to -3 of the Gaussian distribution. The second value of the integral, I_2, would be found by taking the integral from -3 to X_2 and adding to the first value. X_2 is the second value of x, as x ranges over 1001 values from -3 to 3, and would be -2.994. In general, the ith value of the integral in terms of the i-one-th value would be:

$$I_i = I_{i-1} + \int_{xi-1}^{xi} \frac{1}{\sqrt{2\pi}} \exp\left[\frac{-Z^2}{2}\right] dz \qquad (17)$$

For each application of subroutine SIMP to this equation by subroutine IGTAB, the region is divided into 10 partitions; for 1000 applications of subroutine SIMP, the accuracy is more than sufficient and exceeds 13 decimal digits.

Subroutine STORIT

This subroutine stores, in the two-dimensional Fortran array ED, the full range of all the edge profiles to be used in a given test run of program SIMU1D. The values are computed by either interpolating from or picking off uniformly spaced samples out of the lookup table GI. If, for example, the width of the desired edge is 21 pixels and the amount by which subroutine FINDEJ will vary the endpoints in and out to find the best fit signal is 5 units, the original ramp has the endpoints of its Gaussian rise at pixels 41 and 61. As they are varied, the narrowest profile produced has its endpoints at 46 and 56, being 11 pixels wide; and the widest profile produced has its endpoints at pixels 36 and 66, being 31 pixels wide. Subroutine STORIT, therefore, stores 21 Gaussian ramp profiles ranging in width from 11 through 31 units, one profile in each of the 21 rows of the 101-unit length Fortran array ED, the smallest

one first. The procedure of calculating the profiles once and storing them in an array for easy lookup is much more efficient than calculating them repetitively on demand whenever they are needed for analysis.

Subroutine LINTERP

Subroutine LINTERP is used by subroutine STORIT to calculate from the lookup table IGTAB, either by direct data point selection or by interpolation between existing data points, the values of the Gaussian ramps of various widths that are needed. The variable NP is the number of divisions between the input data point in the input array DATA1, and NC, which must be less than NP, is the number of divisions between the points to be set up in the output data array DATA2. Point selection and interpolation, which are used when the variable IFLAG is or is not equal to 0 respectively, are best illustrated by example.

Suppose that the data in the input array have the values 0.0, 0.1, 0.2, ..., 1.0, i.e., 11 values with 10 divisions or steps between them. If one is required to produce a six-point ramp with five divisions between points to represent these data, the values would simply be sampled or selected and are: 0.0, 0.2, 0.4, ..., 1.0. If a four-point representation of the data with three steps between them is required, linear interpolation must be used, resulting in the values 0.0, 0.33, 0.67, and 1.0. The criterion which subroutine LINTERP utilized to decide between point sampling and linear interpolation, therefore, is to check whether the number of points less one in the data set to be produced is evenly divisible into the number of points less one of the input data. If the remainder is 0, the proper samples can simply be read off. If there is a remainder, linear interpolation between the input array points is utilized to construct the Gaussian ramps of the various widths required by the simulation program.

Subroutine SETEJ

In subroutine CALC, when an edge profile of width 21, for example, is first read into the working array EJ, it occupies the 21 leftmost pixel positions. Subroutine SETEJ shifts this profile to lie between two specified left and right coordinates for example, 41 and 61. The region to the left of the profile is padded with 0's and the region to the right with 1's producing an edge as in figure 7.

Subroutine FILLEJ

This subroutine scales the edge ramp as in equation 13 where the particular values of V_s and V_n are read in as the Fortran variables VSIG and VNOI. The subroutine BOXNO, which exists on the ARRADCOM CDC-6000 computer system and which subroutine FILLEJ calls, produces two 0-mean, standard deviation-1, Gaussian random noise values A and B of which the first A is used.

Subroutine MD1DW

This subroutine performs a one-dimensional median filter operation on the values in the array EJ. The number of values read into EJ must not exceed 101, which is the maximum dimensioned space allowed. Also the window width NW must be an odd integer for successful operation.

Subroutine DSORT

This subroutine is used by subroutine MD1DW and will sort out a list of real numbers up to 19 entries. The technique used is a quick-and-efficient compare-and-exchange drop sort.

Subroutine FINDEJ

This routine finds the best edge fit (the one with the lowest mean square error) to the filtered data and delivers its left and right coordinates and the value of the mean square error as its outputs. The order in which the possible ramps are inspected is again best described by example.

If the endpoints of an edge profile, located at pixels 41 and 61 are allowed to vary in and out by an amount v = 5, the left edge will run over the values 36 through 46, and the right edge over values 56 through 66. There are 4v + 1 or 21 possible different widths for the ramps 11 through 31 units. The ramps with the lowest 2v + 1 or 11 widths, that is, 11 through 21 are inspected in the first of two passes and the rest of the Gaussian ramps are checked on the second pass. In the first pass, the first ramp checked is of width 11, the one with endpoints at 46 and 56. Then the ramps of width 12, specified by pixels 46 and 57, and 45 and 56 are checked. Next come the three ramps, 13 units wide, namely. 46 and 58, 45 and 57, and 44 and 56. Following this come 46 and 59, 45 and 58, ..., down to 43 and 56, that is, ramps of width 14. This procedure continues in pass one until the last set checked has ramps of width 21, 46 and 66, 45 and 65, 44 and 64, down to 36 and 56.

In pass 2, the first set of ramps checked has a width 22, for example, 45 and 66, 44 and 65, down to 36 and 57. The procedure continues, and the ramp width is increased. When it is 30, the choices are 37 and 66, and 36 and 65. The last ramp checked has width 31, and left and right endpoints of 36 and 66.

The largest possible value for the mean square error between a pure signal trial ramp and a noisy edge would occur if the signal pixels had the value GRAMAX and the noisy edge had the value -GRAMAX across the board of 101 pixels. The value would be:

$$\begin{aligned} MSE &= \sum_{i=1}^{101} \Big[GRAMAX - (-GRAMAX) \Big]^2 \\ &= 404 \ (GRAMAX)^2 \end{aligned} \qquad (18)$$

This is the initialization value of the Fortran variable ERVAL in which the lowest mean square error will eventually be stored. As each ramp is inspected, if the value of the mean square error is lower than the value currently sitting in ERVAL, ERVAL assumes the value of the mean square error of that ramp and the left and right pixel location coordinates of the ramp endpoints are noted. After all of the 121 ramps have been inspected, the left and right ends of the best fit are known, and its mean square error lies in the variable ERVAL.

Subroutine MSE

This subroutine is rather straightforward and simply calculates the total square error, that is, the sum of the squares of the point-by-point differences of two arrays dimensioned to a length of 101 pixels each.

Subroutine DOMORE

As mentioned in the description of subroutine CALC, this is the subroutine that performs, for each input noise data point, any subsequent runs besides the first if they have been requested by the user for the purpose of eliminating, by averaging, any statistical fluctuations in the mean square error of the output noise. If, for example, 10 runs per data point had been specified, the first is done in subroutine CALC. Then subroutine DOMORE is called on to set up a noisy edge, operate on it with a median filter, and find the best fit ramp and its mean square error for nine different sets of noise samples.

A total of nine mean square figures are passed out of the subroutine in the variable VAL. All 10 of the mean square error values are then averaged in subroutine CALC to produce a value of output noise relatively free of strong individual fluctuations.

Display Subroutines

The last five subroutines handle the display of the output data. Subroutine print 7 uses a Fortran write of the form write (7,100) to write the input and output noise intensity percentages to a local CDC SCOPE file, TAPE7, for storage and availability for further study. Subroutine MINMAX simply notes the minimum and maximum values of the output noise intensities in the array Y. Subroutine PRINOU produces, if it has been requested, a chart of the input and output noise data. As explained in detail in the section on subroutine READIN, subroutine MARGIN prepares the output ploting area, leaving designated bordering portions of it blank and compressing the plot of the results into the central region of the plotting area. Subroutine PLOT plots the output versus input noise intensities up to 1001 points of data. The graphics commands assume a Tektronix 4014 terminal.

PROGRAM LOCKON

To explore and properly interpret the preliminary results of program SIMU1D, the development of program LOCKON became necessary. For example, for 10% Gaussian noise intensity, the best fit signal ramp, after median filtering, did not agree with the input signal ramp a good percentage of the time. To statistically measure how often mismatch occurs for a given percentage noise intensity and edge, and window widths, program LOCKON was devised.

The program first asks for the input edge and window widths, and the number of trial runs per data point. Finally the user furnishes the minimum and maximum on the range of noise intensity percentages and the number of data points to be tested. The program then proceeds to calculate three quantities to be explained, P_B, P_A, and P_{DIF}.

In a manner similar to the one-dimensional simulation, Program LOCKON sets up an input edge, applies a median filter to it, and finds the best fit signal with the lowest mean square error. Specifically, when a noisy edge F(t) is set up, subroutine FINDEJ is called upon to find the left and right coordinates of the best fit edge ramp. If either coordinate does not coincide with those of the input signal, a mismatch is noted. The prescribed median filter is then applied, yielding G(t). The best fit for these data is then found by subroutine FINDEJ; and any mismatch with the input signal is again noted. This process is repeated for each data point for however many runs the user has specified. For a given number N of runs, the percent of time in which the noisy signal, before filtering, did not match the input signal is the miss probability P_B. The percent of time the noisy edge, after median filtering, did not match the input signal ramp coordinates is the miss probability P_A. The percent of time the signals before and after application of the median filter differed from each other is the miss probability P_{DIF}.

PROGRAMS EJSHOW AND PULPLT

These are graphics programs developed by the investigator to help conceptualize and visualize the kinds of results being generated by the median filtering windows being used in the one-dimensional simulation program SIMU1D.

Program EJSHOW

Program EJSHOW will graph a noisy edge F(t) and two signals. One signal will be either the best fit signal or another user-specified signal; and a second signal, will be the user's choice, for example, a poor ramp fit for purposes of comparison.

The program will also report the output noise percentage intensities on these signal choices before and after application of the desired one-dimension median filtering window being used.

Program PULPLT

Program PULPLT was developed to plot the noisy square wave pulses before and after median filtering, as shown in figures 3 and 4. The input pulse width is specified by the user as a percentage of the total available plotting field. The four Fortran program/modular subroutine simulation packages, SIMU1D, LOCKON, EJSHOW, and PULPLT, are listed in appendixes A, B, C, and D, respectively, at the end of this report.

PRELIMINARY RESULTS

Figure 8 is a plot of the kind of results that program SIMU1D will generate. A total of 101 points are plotted for input noise intensity percentage ranging from 0 to 100%. The ordinate, the postprocessing noise rises from 0 to 65%. This graph incorporates an edge ramp of width 21 and a median filter window width of 3. As the percentage of noise P_n rises on the graph, the corresponding percentage of signal $(1 - P_n)$ diminishes. This becomes apparent to the reader, for the high noise portions of the graph, as the smoothness of the curve diminishes and disappears into the completely uncorrelated and predominating Gaussian noise.

Figure 9 shows an example of a noisy one-dimensional edge profile F(t) with 10% Gaussian noise intensity for a Gaussian ramp having end-points 41 and 61. Also shown are two examples of pure signals: the best fit signal is on the right having a Gaussian ramp rising from pixels 41 through 61; an example of a poor fit signal is on the left curve with a ramp rising from points 36 through 56. Figure 10 is an example of the profile G(t) obtained by filtering the noisy input F(t) of figure 9 with a median filter window of width 3. Again, for comparison purposes, two pure signals are shown, the best fit to the right, with a Gaussian ramp rising from pixels 41 through 61, and the poor fit to the left with a Gaussian ramp rising from pixels 36 through 56.

In addition to these kinds of results, preliminary tests with program LOCKON were conducted on the 10% input noise percentage intensity edges. For 100 trial runs, for example, the pre- and postfiltered edge ramps do not match the input signal 40% of the time and do not match each other about 20% of the time out of 100. These kinds of results must be further developed, interpreted, and extended to the analysis of two-dimensional simulated data before any testing of combined median filter window outputs upon actual digitized imagery data.

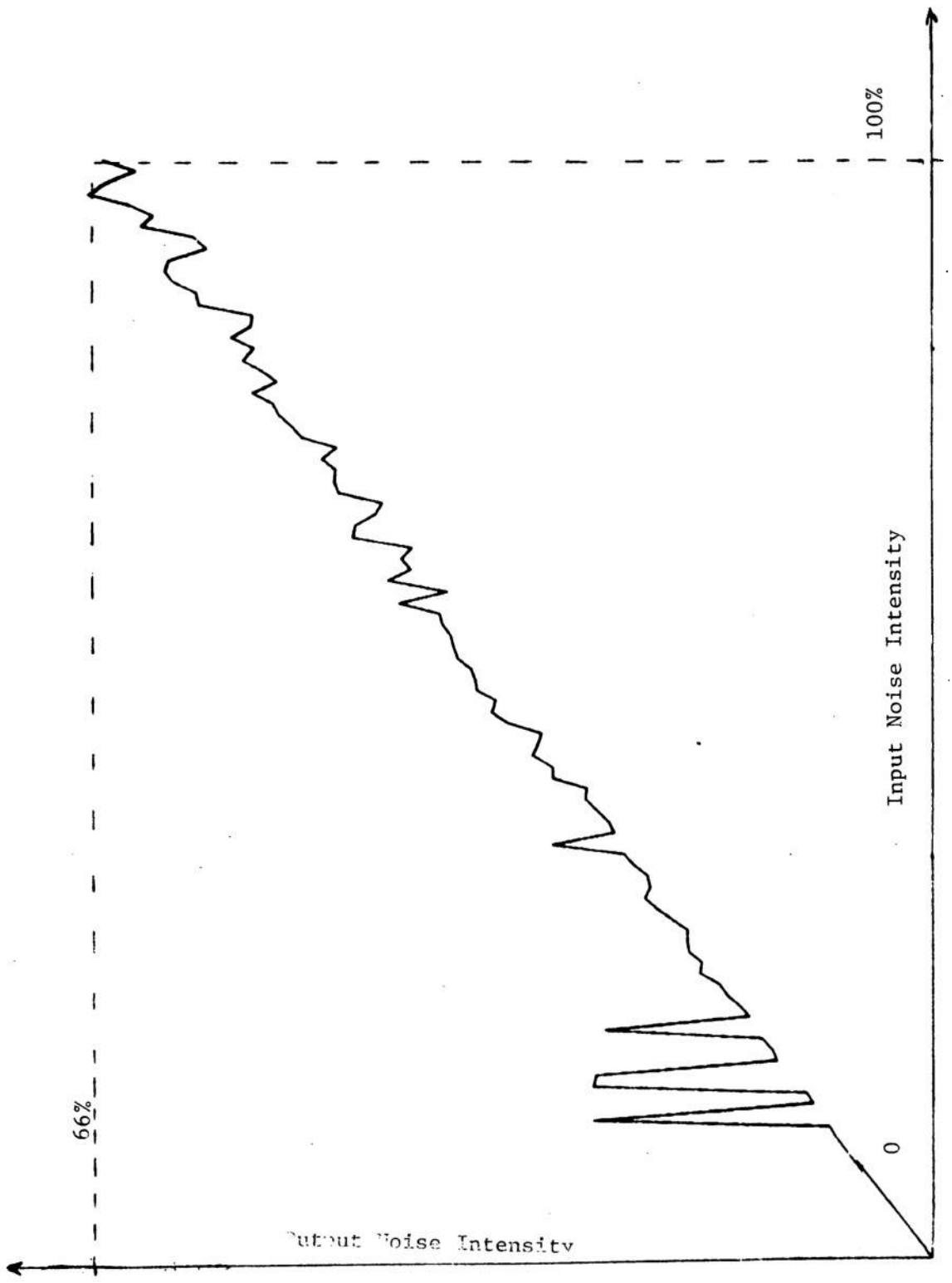

Figure 8. Filtered versus input Gaussian noise: one-dimensional simulation

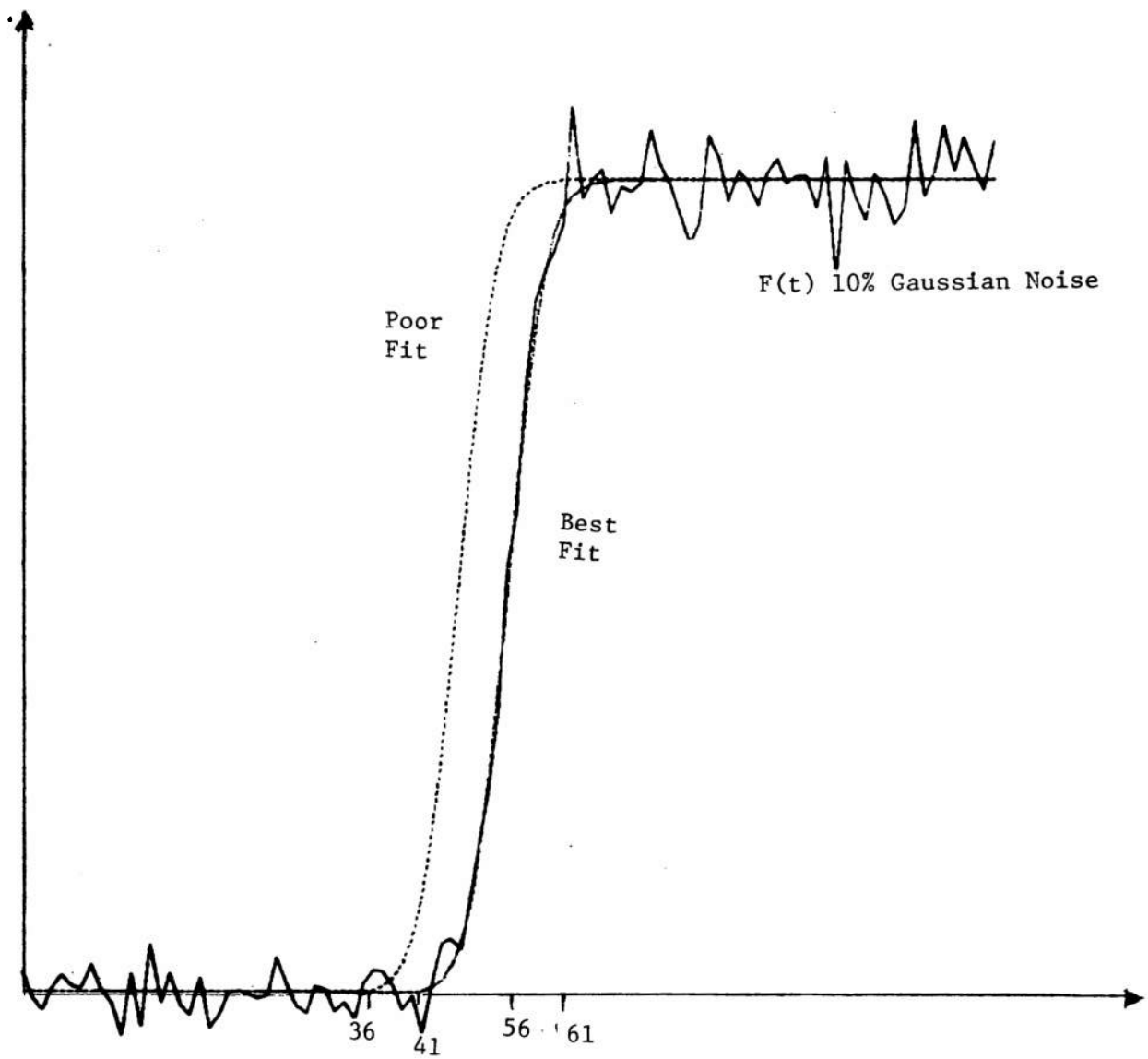

Figure 9. One-dimensional noisy edge

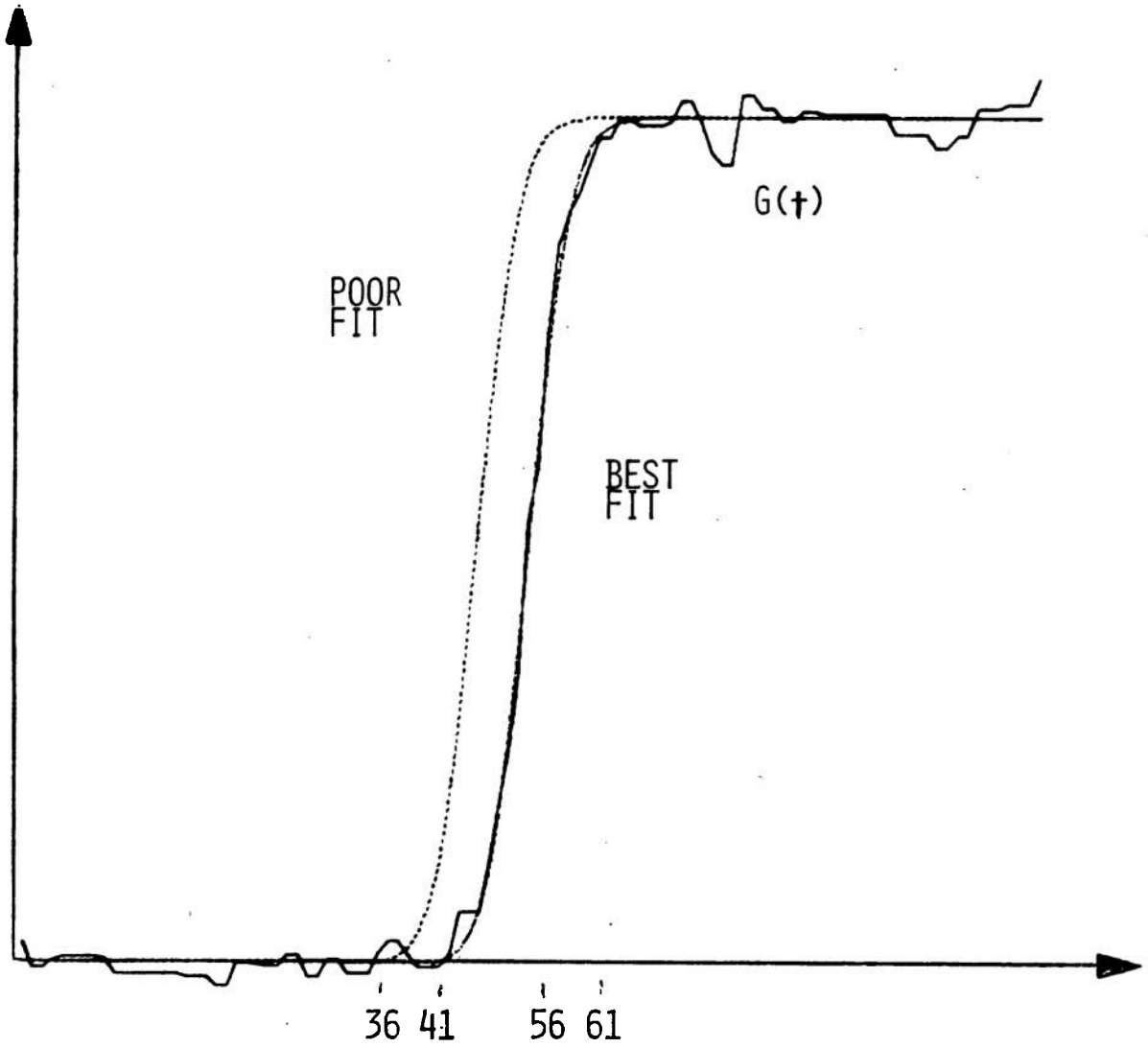

Figure 10. One-dimensional edge analysis, 10% Gaussian noise, filtering window width 3

RECOMMENDATIONS

Additional software must be developed to properly interpret the type of results being produced by programs SIMU1D and LOCKON. First, in terms of viewing digital imagery of FLIR or visible TV-displayed signals, 40 dB is considered good in practice. Any noise intensity greater than 3%, that is, a signal-to-noise ratio 15 dB or less is considered severe. The noise intensities simulated do not go much beyond 3% and definitely do not exceed 5%, that is, less than 13 dB. Thus, a study of the effect of median filters on noise level intensities up to 5% (signal-to-noise ratios greater than 13 decibels) will indicate the multiplicative factor or scaling function by which particular median filters will supress noise.

The mismatch probabilities between the input signal and the noisy edges before and after filtering, as produced by program LOCKON, are insufficient data from which to draw conclusions about any possible degradations of edges by median filters. Two approaches are suggested: They must be assessed and a choice must be made on the desirability of one of them.

First, instead of computing the mismatch probabilities between the input signal and the noisy ramp, or the input signal and the median filtered ramp, program LOCKON should be modified to produce, for a given number N of trial runs, the means and the standard deviations of the locations of the left and right coordinates of the endpoints of the Gaussian ramps. These quantities are more useful than the mismatch probabilities for determining the effects of median filter windows on edges and can be used to obtain measures of the mean edge width and the standard deviation of how much it tends to vary. The standard deviation of variance of the edge width will give some indication of how much a particular median filter might be degrading the given edge.

The question arises as to what is the minimum number N of trial runs required for a good standard deviation measurement. Assume that a pure signal ramp, scaled to the maximum value GRAMAX, contains 101 points P_I. A median filtered noisy edge, whose best fit, pure signal is assumed not to match the signal P_I, contains 101 points Q_I. The root mean square separation of the two curves is:

$$\Delta = \sqrt{\frac{(Q_j - P_j)^2}{101}} \tag{19}$$

If one scales out the GRAMAX dependence, the mean square percentage error between these two curves is:

$$MSE = \frac{100\ \Delta}{GRAMAX} \tag{20}$$

To calculate the number of trial runs required for a good standard deviation measurement of the Gaussian ramp endpoints, the average noise produced must be much less than the shift Δ of equation 19. If the noise is on the order of or greater than the Δ shift between the two curves, that shift and, therefore, the degradation produced by the given median filter will be masked by the noise. The noise of one test run is the value V_n of equation 13, which for N runs goes down by the square root of N. One therefore has:

$$\frac{V_N}{N} \ll \Delta \tag{21}$$

For a factor a $<$ one, which is used to ensure that the number N of trial runs will be high enough so that the Gaussian noise does not mask the degradations produced by a given median filter, one has from equations 13, 20 and 21:

$$\frac{\left(\frac{P_N}{100}\right)\left(\frac{GRAMAX}{SIGMAS}\right)}{\sqrt{N}} < \frac{(a)\ (MSE)\ (GRAMAX)}{100} \tag{22}$$

This reduces to:

$$N > \left(\frac{P_N}{MSE}\right)^2 \left(\frac{1}{a\ (SIGMAS)}\right)^2 \tag{23}$$

For a masking prevention factor a = 0.5 and SIGMAS = 3 standard deviation

$$N > 0.44\left(\frac{P_n}{MSE}\right)^2 \tag{24}$$

This yields, say for 3% noise and 0.3% mean square error shift between the noisy and pure signal curves, an N of at least 44 trial runs for a legitimate measure of the standard deviation or variance of the Gaussian ramp edges.

A second approach is available for detecting noise degradations of one-dimensional Gaussian edge ramps that is far more simple. Instead of finding the best fits for the noisy edge ramp with separate sets of noise points and noting the average and standard deviation of these fits, add up a series of N noisy edge ramps and average them immediately. The noise will then decrease as $\frac{1}{N}$. For example, adding up 100 ramps with identical signals and different noise samples will cut down the noise by a factor of 10.

Inspection of the resulting average ramp by subroutine FINDEJ will yield the best fit signal and information as to any degradations the median filter has produced without the problem of strong potentially masking the effect of small edge degradations.

These studies are easily extended to the investigation of the effects of two-dimensional median filters on an edge in, for example, a 101 by 101 area. Each horizontal line of the synthetic picture array is formed by repeating the edge ramp slice E(t), thus forming a vertical edge line in the picture. Gaussian noise of a given percentage intensity is then added and the whole frame is operated on by the desired two-dimensional median filter. Each line of the synthetic picture can then be inspected by subroutine FINDEJ to find the best fit pure signal. If the edge line formed by the centers of the 101 input signal ramps is vertical, the average and standard deviations of these 101 ramps will then give an idea of how the two-dimensional median filter may be degrading the edge. Alternatively, if the edge line of the ramp is not necessarily vertical, the location of the centers of the edge ramps in each horizontal line of the picture can be noted and the best line-fit through them can be calculated and compared with the orginal input edge line. Investigation of edges oriented at various angles within the picture field will then determine if two-dimensional median filter windows possess any inherent orientational biases toward the edges surrounding targets of interest in a given picture.

Once the action of two-dimensional median filters has been fully understood and, if acceptable, the author suggests tests on actual imagery data as in equations 7 through 10. Here θ and $\bar{d}$ can vary separately or together to produce different median filter window combinations governed by continuous functions of the picture gradient for optimal noise suppression and minimal, or no, signal degradation.

REFERENCES

1. James W. Tukey, Exploratory Data Analysis, Reading, MA: Addison-Wesley, 1971 and 1979.

2. William K. Pratt, Digital Image Processing, John Wiley and Sons, 1978, p 333.

3. Professor Thomas S. Huang, Dept. Electrical Engineering, Purdue, Lafayette, Indiana (private communication).

BIBLIOGRAPHY

Andrews, Harry C., and John Baird Morton, "A Posteriori Method of Image Restoration," Journ. Optical Society of America, February 1979, pp 280-290.

Andrews, Harry C., Digital Image Restoration, Prentice Hall, 1976.

Castleman, Kenneth R., Digital Image Processing, Prentice Hall, 1979.

Froehlich, Gary K., Robert B. Asher, and John F. Walkup, "Optical Estimation in Signal-Dependent Noise," JOSA, December 1978, pp 1665-1671.

Gonzalez, Raphael C. and Paul A. Wintz, Digital Image Processing, Reading, MA: Addison-Wesley, 1977.

Huang, T.S. and G. Yang, "A Fast Two-Dimensional Median Filtering Algorithm," IEEE Transactions on Acoustics, Speech, and Signal Processing, February 1979, pp 13-18.

Jayant, N.S., "Average and Median-Base Smoothing Techniques for Improving Digital Speech Quality in the Presence of Transmission Errors," IEEE Transactions Communications, September 1976, pp 1043-1045.

Justusson, B., "Statistical Properties of Median Filters in Signal and Image Processing," Unpublished report., Math. Institute, Royal Institute of Technology, Stockholm, Sweden, December 1977.

Kanefsky, Morton and Michael Strintzis, "A Decision-Theory Approach To Picture Smoothing," IEEE Transactions on Computers, January 1978, pp 32-38.

Marmolin, H., S. Nyberg and U. Berggrund, "A Visual Optimized Restoring Filter," FOA report C56016-H9, December 1977, National Institute for Defense Research, Sweden.

Naderi, Sawchuk, "Detection of Low-Contrast Images in Film-Grain Noise," Applied Optics, September 15, 1978, pp 2883-2891.

Panda, Durga, "Recursive Least-Squares Smoothing of Noise in Images," with A.C. Kak, IEEE Transactions on Acoustics, Speech, and Signal Processing, December 1977, pp 520-524.

Pratt, William K., "Median Filtering, Semiannual Report," Image Processing Institute, University Southern California, September 1975, pp 116-123.

Rabiner, L.R., M.R. Sambur, C.E. Schmidt, "Application of a Nonlinear Smoothing Algorithm To Speech Processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, December 1975, pp 552-557.

Rosenfeld, A. and A.C. Kak, Digital Picture Processing, Academic Press, 1976.

Schreiber, W.F., "Image Processing for Quality Improvement," IEEE Proceedings, December 1978, pp 1640-1651.

Zweig, Barrett, and Hu, "Noise-Cheating Image Enhancement, JOSA, November 1975, pp 1347-1353.

APPENDIX A

PROGRAM SIMUID

```
      PROGRAM SIMU1D(INPUT,OUTPUT,TAPE7,TAPE61=100,TAPE62=100)
      DIMENSION X(1001),Y(1001)
C-TAKES A ONE-DIMENSIONAL SLICE OF AN IMAGE EDGE RAMP SIMULATED BY THE
C-INTEGRAL OF A GAUSSIAN, ADDS NOISE, MEDIAN FILTERS IT, MEASURES
C-THE NEW NOISE, AND PLOTS OUTPUT VS. INPUT % NOISE.
      CALL CONNEC(5LINPUT) $ CALL CONNEC(6LOUTPUT)
      CALL READIN(DISP,NE,NW,NRUN,XMIN,XMAX,NP,XLM,RM,TM,BM)
      CALL CALC(NE,NW,NRUN,X,Y,NP,DX,XMIN,XMAX)
      CALL PRINT7(X,Y,NP)
      CALL MINMAX(Y,NP,YMIN,YMAX)
      PRINT *,"OUTPUT NOISE %S: MIN = ",YMIN,", MAX = ",YMAX
      FUZZ = .0001*DX
      YMAX = YMAX + FUZZ $ YMIN = YMIN - FUZZ
      IF ( DISP .EQ. "PRINT") GO TO 1
      CALL MARGIN(XMIN,XMAX,YMIN,YMAX,XLM,RM,TM,BM)
    1 IF(DISP.EQ."PLOT") GO TO 2
      CALL PRINOU(X,Y,NP)
    2 IF(DISP.EQ."PRINT") GO TO 3
      CALL PLOT(X,Y,NP,XMIN,XMAX,YMIN,YMAX)
    3 STOP
      END
      SUBROUTINE READIN(DISP,NE,NW,NRUN,XMIN,XMAX,NP,XLM,RM,TM,BM)
    1 PRINT *,"WHICH DISPLAY, PRINT, PLOT, OR BOTH? "
      READ 100,DISP
  100 FORMAT(A5)
      IF(DISP.EQ."PRINT".OR.DISP.EQ."PLOT".OR.DISP.EQ."BOTH") GO TO 2
      PRINT *,"INCORRECT DISPLAY TYPE."
      GO TO 1
    2 PRINT*,"TYPE EDGEWIDTH, MEDIAN FILTER WINDOW SIZE, & NO. RUNS. "
      READ *,NE,NW,NRUN
      IF(NE .GE. NW .AND. MOD(NW,2) .EQ. 1) GO TO 3
      IF ( NE .LT. NW ) GO TO 4
      PRINT *,"WINDOW SIZE ",NW," MUST BE AN ODD NO."
      GO TO 2
    4 PRINT *,"EDGE SIZE ",NE," MUST BE AT LEAST WINDOW SIZE ",NW
      GO TO 2
    3 PRINT *,"ENTER MIN&MAX INPUT NOISE %, & NO.POINTS NOT .GT. 1001 "
      READ *,XMIN,XMAX,NP
      IF(DISP.EQ."PRINT") GO TO 5
      PRINT *,"ENTER LEFT, RIGHT, UPPER & LOWER MARGIN PORTIONS. "
      READ *,XLM,RM,TM,BM
      GO TO 6
    5 XLM = RM = TM = BM = 0.0
    6 RETURN
      END
      SUBROUTINE CALC(NE,NW,NRUN,X,Y,NP,DX,XMIN,XMAX)
      DIMENSION GI(1001),ED(21,101),X(1001),Y(1001),EJ(101)
C-GIVEN EDGE & WINDOW SIZES, & RANGE ON INPUT NOISE %S,
```

```
C-THIS ROUTINE CONTROLS THE CALCULATIONS OF THE OUTPUT NOISE %S.
C-SETUP TABLE OF INTEGRAL OF GAUSSIAN FROM XMIN TO X OF EXP(-X*X/2)
      CALL IGTAB(GI)
      NVAR = 5
      CALL STORIT(GI,ED,NE,NVAR)
      NL1 = (101-NE)/2+1 $ NR1 = (101+NE)/2
      IBASE = NE-2*NVAR-1 $ IRAMP = NE-IBASE
      CALL SECOND(A) $ NTIME = 1000*(A+1.0) $ CALL RDMIN(NTIME)
C-USES COMPUTER TIME TO INSURE RANDOM NO. GENER. NOT START IN SAME SPOT.
      GRAMAX = 255. $ SIGMAS = 3.0
C-MAX GRAY LEVEL TO HAVE, & NO. STANDARD DEVIATIONS.
      DX = (XMAX - XMIN ) / FLOAT(NP-1)
      I = 0
    1 I = I+1
      X(I) = XMIN + DX*FLOAT(I-1)
      VNOI = X(I)/100.*GRAMAX/SIGMAS
      VSIG = (1.-X(I)/100.)*GRAMAX
C-USED TO SCALE SIGNAL & NOISE SO GRA LEVELS SHOULD NOT EXCEED GRAMAX.
C-SETUP NOISY EDGE, EJ.
      DO 2 J = 1,NE
    2 EJ(J) = ED(IRAMP,J)
      CALL SETEJ(NL1,NR1,EJ)
      CALL FILLEJ(VSIG,VNOI,EJ)
      CALL MD1DW(NW,EJ)
C-MEDIAN FILTERS NOISY EDGE.
      CALL FINDEJ(NE,NL1,NR1,NVAR,ED,EJ,VSIG,GRAMAX,ERVAL,NL2,NR2)
      ERVAL = SQRT(ERVAL/101.0)
C-FINDS BEST EDGE FIT TO FILTERED DATA, (ONE WITH LOWEST MEAN SQUARE
C-ERROR, & GIVES LEFT & RIGHT HAND ENDPOINT COORDINATES.
      IF(NRUN .EQ. 1) GO TO 3
      CALL DOMORE(NL2,NR2,NVAR,ED,EJ,VSIG,VNOI,NW,NRUN,VAL)
C-CONDUCTS MORE NOISE RUNS ON THE SOUGHT-FOR EDJE.
      ERVAL = ERVAL + VAL
      ERVAL = ERVAL / NRUN
    3 Y(I) = 100.*ERVAL*SIGMAS/GRAMAX
      IF ( I .LT. NP ) GO TO 1
      RETURN
      END
      SUBROUTINE IGTAB(GI)
      DIMENSION GI(1001)
C-SETSUP TABLE OF INTEGRAL OF GAUSSIAN, FROM XMIN TO X OF EXP(-X*X/2)
      PRINT *,"DOING INTEGRAL GAUSSIAN TABLE."
      XMIN = -3.0 $ XMAX = 3.0
      NP = 1001 $ NDIV = 10
      DX = (XMAX-XMIN) / FLOAT(NP-1)
      CONST = 1./SQRT(8.*ATAN(1.))
      XOLD = XMIN $ GI(1) = 0.0
      I = 0
    1 I = I+1
      X = XMIN + DX*FLOAT(I-1)
      IF ( I .EQ. 1 ) GO TO 2
```

```
        CALL SIMP(XOLD,X,NDIV,ANS)
        GI(I) = GI(I-1) + CONST*ANS
        XOLD = X
      2 IF ( I .LT. NP ) GO TO 1
        PRINT *,"TABLE DONE, PROCEEDING TO SIMULATION. "
        RETURN
        END
        SUBROUTINE SIMP(XMIN,XMAX,N,ANS)
C-INTEGRATES A FUNCTION BY SIMPSON'S RU.E
        H = (XMAX - XMIN ) / FLOAT(N)
        NODD = N-1
        NEVEN = N-2
        ODSUM = EVSUM = 0.0
        IODD = -1
        IEVEN = 0
      1 IODD = IODD + 2
        X = XMIN + IODD*H
        Y = F1(X)
        ODSUM = ODSUM + Y
        IF ( IODD .LT. NODD ) GO TO 1
      2 IEVEN = IEVEN + 2
        X = XMIN + IEVEN * H
        Y = F1(X)
        EVSUM = EVSUM + Y
        IF ( IEVEN .LT. NEVEN ) GO TO 2
        YMIN = F1(XMIN)
        YMAX = F1(XMAX)
        ANS = H/3.0 * (YMIN + 4.0*ODSUM + 2.0*EVSUM + YMAX)
        RETURN
        END
        FUNCTION F1(X)
        F1 = EXP(-X*X/2.0)
        RETURN
        END
        SUBROUTINE STORIT(GI,ED,NE,NVAR)
        DIMENSION GI(1001),X(101),ED(21,101)
C-THIS INTERPOLATES THE SET OF EDGE RAMPS TO BE USED LATER.
        IF(NVAR .LE. 10) GO TO 1
        PRINT *,"NVAR, ",NVAR,", SHOULD BE 10 OR LESS."
        STOP
      1 IBASE = NE-2*NVAR-1
        NRAMPS = 4*NVAR+1
        DO 2 IRAMP = 1,NRAMPS
        NWIDE = IBASE + IRAMP
        CALL LINTERP(GI,X,1001,NWIDE)
        DO 2 J = 1,NWIDE
      2 ED(IRAMP,J) = X(J)
        RETURN
        END
        SUBROUTINE LINTERP(DATA1,DATA2,NP,NC)
        DIMENSION DATA1(1001),DATA2(101)
```

```
C-PICKS OR LINEARLY INTERPS. BIG DATA SET TO GET SMALLER ONE.
      SKIP = FLOAT(NP-1) / FLOAT(NC-1)
      IFLAG = MOD( (NP-1), (NC-1) )
      IF ( IFLAG .NE. 0 ) GO TO 1
C-INTERP BY PICKING OFF POINTS, E.G. EVERY 10TH OF 1000.
      IC = 0
    2 IC = IC + 1
      JC = SKIP*FLOAT(IC-1)+1
      DATA2(IC) = DATA1(JC)
      IF ( IC .LT. NC) GO TO 2
      GO TO 3
C-INTERP. BY PICKING LINEARLY BETWEEN POINTS.
    1 IC = 0
    4 IC = IC+1
      XVAL = SKIP*FLOAT(IC-1) +1
      I1 = XVAL $ Q = XVAL-I1
      DATA2(IC) = DATA1(I1) + Q*(DATA1(I1+1)-DATA1(I1))
      IF ( IC .LT. NC) GO TO 4
    3 RETURN
      END
      SUBROUTINE SETEJ(NLEFT,NRIGHT,EJ)
      DIMENSION EJ(101)
C-MAKES INTERPED. GAUSIAN RAMP RISE FROM 0 TO 1 BETWEEN NLEFT & NRIGHT.
      NWIDE=NRIGHT-NLEFT+1$NWIDE1=NWIDE+1$ISHIFT=NLEFT-1$IPAST=NRIGHT+1
C-TRIAL WIDTH, 1 MORE, NO.POINTS TO SHIFT TO CENTER, 1 PAST RIGHT END.
      DO 1 I = IPAST,101
    1 EJ(I) = 1.0
      DO 2 I = 1,NWIDE
      J = NWIDE1-I $ K = J+ISHIFT
    2 EJ(K) = EJ(J)
      DO 3 I = 1,ISHIFT
    3 EJ(I) = 0.0
      RETURN
      END
      SUBROUTINE FILLEJ(VSIG,VNOI,EJ)
      DIMENSION EJ(101)
C-SCALES EDGE TO MAX OF GRAMAX WITH SIGNAL & NOISE CONTRIBUTIONS.
      DO 1 I = 1,101
      CALL BOXNO(A,B)
C-GAUSSIAN RANDOM NOISE GENER. GIVES 2 USABLE VALUES.
C-NOW SCALE SIGNAL BY VSIG AND ADD NOISE TIMES VNOI.
    1 EJ(I) = VSIG*EJ(I) + VNOI*A
      RETURN
      END
      SUBROUTINE MD1DW(NW,EJ)
      DIMENSION EJ(101),SORT(19)
C-ONE-D MEDIAN FILTERS EJ.
      MDN1 = NW/2 $ MDN = MDN1+1
      NDO = 101-NW+1
C-NO. OF APPLICATIONS OF WINDOW.
      DO 1 I = 1,NDO
```

```
      DO 2 J = 1,NW
      K = I + (J-1)
    2 SORT(J) = EJ(K)
      CALL DSORT(SORT,NW)
      L = I + MDN1
    1 EJ(L) = SORT(MDN)
      RETURN
      END
      SUBROUTINE DSORT(X,N)
      DIMENSION X(19)
C-DOES A DROP SORT OF UP TO 19 NUMBERS.
      I = 1
    1 IF(X(I) .GT. X(I+1) ) GO TO 2
    3 I = I+1
      IF ( I .LT. N) GO TO 1
      GO TO 4
    2 CALL SWAP( X(I), X(I+1) )
      J = I-1
    5 IF(J .EQ. 0 ) GO TO 3
      IF( X(J) .GT. X(J+1) ) CALL SWAP(X(J), X(J+1) )
      J = J-1
      GO TO 5
    4 RETURN
      END
      SUBROUTINE SWAP(X,Y)
      Z = X $ X = Y $ Y = Z
      RETURN
      END
      SUBROUTINE FINDEJ(NE,NL1,NR1,NVAR,ED,EJ,VSIG,GRAMAX,ERVAL,NL2,NR2)
      DIMENSION ED(21,101),EJ(101),EJSIG(101)
C-FINDS BEST EDGE FIT TO FILTERED DATA, (ONE WITH LOWEST MEAN SQUARE
C-ERROR), AND GIVES LEFT AND RIGHT HAND END COORDINATES.
      IF(NE .LT. (2*NVAR+2) ) NVAR = (NE-2)/2
      IF(NE .GT. (101-2*NVAR) ) NVAR = (101-NE)/2
      IBASE = NE-2*NVAR-1
      ERVAL = 4.0*101.0*GRAMAX**2
      IPASS = 0
    1 IPASS = IPASS + 1
      IF(IPASS .EQ. 2) GO TO 2
      IMIN=1 $ IMAX=2*NVAR+1 $ NTIME=0 $ SIGN=1.0
      GO TO 3
    2 IMIN=2*NVAR+2 $ IMAX=4*NVAR+1 $ NTIME=2*NVAR+1 $ SIGN=-1.0
    3 DO 4 IRAMP = IMIN,IMAX
      NTIME = NTIME+SIGN*1.0
      NWIDE = IBASE + IRAMP
      IF(IPASS .EQ. 1) NLWORK = NL1 + NVAR
      IF(IPASS.EQ.1) NRWORK = NLWORK + NWIDE -1
      IF(IPASS .EQ. 2) NRWORK = NR1+NVAR
      IF(IPASS .EQ. 2) NLWORK = NRWORK-NWIDE+1
      ITIME = 0
    5 ITIME = ITIME + 1
```

```
      DO 6 J = 1,NWIDE
    6 EJSIG(J) = ED(IRAMP,J)
      CALL SETEJ(NLWORK,NRWORK,EJSIG)
      DO 7 J = NLWORK,101
    7 EJSIG(J) = EJSIG(J) * VSIG
      CALL MSE(EJSIG,EJ,ERROR)
C-FINDS MEANSQUARE ERROR OR NOISE OF EDGE OVER PURESIGNAL EJSIG.
      IF(ERROR.GE. ERVAL) GO TO 8
      ERVAL = ERROR
      NL2 = NLWORK $ NR2 = NRWORK
    8 NLWORK=NLWORK-1 $ NRWORK=NRWORK-1
    4 IF(ITIME .LT. NTIME) GO TO 5
      IF(IPASS .LT. 2) GO TO 1
      RETURN
      END
      SUBROUTINE MSE(X,Y,ERROR)
      DIMENSION X(101),Y(101)
      ERROR = 0.0
C-THIS CALCULATES THE MEAN SQUARE ERROR OF Y FROM X.
      DO 1 I = 1,101
    1 ERROR = ERROR + ( Y(I)-X(I) )**2
      RETURN
      END
      SUBROUTINE DOMORE(NLEFT,NRIGHT,NVAR,ED,EJ,VSIG,VNOI,NW,NRUN,VAL)
      DIMENSION ED(21,101),EJ(101),EJSIG(101)
C-NRUN-1 MORE RUNS ON THE FOUND EDGE TO SMOOTH THE NOISE RESULT.
      NWIDE=NRIGHT-NLEFT+1 $ IBASE=NWIDE-2*NVAR-1 $ IRAMP=NWIDE-IBASE
C-GO PICK THE IRAMP ROW OF ARRAY ED.
      DO 1 J = 1,NWIDE
    1 EJSIG(J) = ED(IRAMP,J)
      CALL SETEJ(NLEFT,NRIGHT,EJSIG)
      DO 2 J = NLEFT,101
    2 EJSIG(J) = EJSIG(J) * VSIG
      VAL = 0.0 $ IRUN = 1
    3 IRUN = IRUN +1
      DO 4 J = 1,NWIDE
    4 EJ(J) = ED(IRAMP,J)
      CALL SETEJ(NLEFT,NRIGHT,EJ)
      CALL FILLEJ(VSIG,VNOI,EJ)
      CALL MD1DW(NW,EJ)
      CALL MSE(EJSIG,EJ,ERROR)
      ERROR = SQRT(ERROR/101.0)
      VAL = VAL + ERROR
      IF ( IRUN .LT. NRUN ) GO TO 3
      RETURN
      END
      SUBROUTINE PRINT7(X,Y,NP)
      DIMENSION X(1001),Y(1001)
C-THIS WRITES THE INPUT AND OUTPUT NOISE %S TO TAPE7.
      DO 1 I = 1,NP
    1 PRINT(7,*) X(I),",",Y(I)
```

```
      REWIND 7
      RETURN
      END
      SUBROUTINE MINMAX(Y,NP,YMIN,YMAX)
      DIMENSION Y(1001)
      YMIN = YMAX = Y(1)
      DO 1 I = 1,NP
      IF ( Y(I) .LT. YMIN ) YMIN = Y(I)
    1 IF ( Y(I) .GT. YMAX ) YMAX = Y(I)
      RETURN
      END
      SUBROUTINE MARGIN(XMIN,XMAX,YMIN,YMAX, XLM,RM,TM,BM)
      HH = (XMAX-XMIN) / (1.0-XLM-RM)
      HV = (YMAX-YMIN) / (1.0-TM-BM)
      XMIN = XMIN -HH*XLM $ XMAX = XMAX + HH*RM
      YMAX = YMAX + HV*TM $ YMIN = YMIN - HV*BM
      RETURN
      END
      SUBROUTINE PRINOU(X,Y,NP)
      DIMENSION X(1001),Y(1001)
      READ 100,ICHAR
  100 FORMAT(A1)
      PRINT 200
  200 FORMAT(1H ,"LINE",1X,"X",14X,"Y")
      DO 1 I = 1,NP
      PRINT 300,I,X(I),Y(I)
  300 FORMAT(1H ,I3,2X,2(F14.9,1X) )
      IF ( MOD(I,30) .NE. 0) GO TO 1
      READ 100,ICHAR $ PRINT 200
    1 CONTINUE
      RETURN
      END
      SUBROUTINE PLOT(X,Y,NP, XMIN,XMAX,YMIN,YMAX)
      DIMENSION X(1001),Y(1001)
      READ 100,ICHAR
  100 FORMAT(A1)
      CALL INITT(30) $ CALL TERM(2,4096)
      CALL DWINDO(XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA( X(1), Y(1) )
      DO 1 I = 2,NP
    1 CALL DRAWA( X(I), Y(I) )
      CALL ANMODE $ READ 100,ICHAR
      RETURN
      END
```

APPENDIX B

PROGRAM LOCKON

```
      PROGRAM LOCKON(INPUT,OUTPUT,TAPE7,TAPE61=100,TAPE62=100)
      DIMENSION GI(1001),ED(21,101),X(1001),EJ(101),PB(1001),PA(1001),
     &PDIF(1001)
      CALL CONNEC(5LINPUT) $ CALL CONNEC(6LOUTPUT)
      CALL IGTAB(GI)
      CALL SECOND(A) $ NTIME = 1000*(A+1.0) $ CALL RDMIN(NTIME)
      GRAMAX = 255. $ SIGMAS = 3.0
      PRINT *,"TYPE EJWITH, MEDIANFILTER WINDOWSIZ, & NO.RUNS "
      READ *,NE,NW,NRUN
      NVAR = 5
      IBASE=NE-2*NVAR-1 $ IRAMP = NE-IBASE
      CALL STORIT(GI,ED,NE,NVAR)
      PRINT *,"ENTER MIN&MAX %NOISE, &NO.POINTS NOT.GT. 1001 "
      READ *,XMIN,XMAX,NP
      NLEFT=(101-NE)/2+1 $ NRIGHT=(101+NE)/2
      DX = (XMAX - XMIN ) / FLOAT(NP-1)
      I = 0
    1 I = I+1
      X(I) = XMIN + DX*FLOAT(I-1)
      VNOI = X(I)/100.*GRAMAX/SIGMAS
      VSIG = (1.-X(I)/100.)*GRAMAX
      MA = MB = MDIF = 0
      DO 2 IRUN = 1,NRUN
      DO 3 J = 1,NE
    3 EJ(J) = ED(IRAMP,J)
      CALL SETEJ(NLEFT,NRIGHT,EJ)
      CALL FILLEJ(VSIG,VNOI,EJ)
      CALL FINDEJ(NE,NLEFT,NRIGHT,NVAR,ED,EJ,VSIG,GRAMAX,EB,NLB,NRB)
      CALL MD1DW(NW,EJ)
      CALL FINDEJ(NE,NLEFT,NRIGHT,NVAR,ED,EJ,VSIG,GRAMAX,EA,NLA,NRA)
      IF(NLB.NE.NLEFT.OR.NRB.NE.NRIGHT) MB = MB +1
      IF(NLA.NE.NLEFT.OR.NRA.NE.NRIGHT) MA = MA +1
    2 IF(NLA.NE.NLB .OR. NRA.NE.NRB) MDIF = MDIF +1
      PB(I) = 100.*FLOAT(MB) / NRUN
      PA(I) = 100.*FLOAT(MA) / NRUN
      PDIF(I) = 100.*FLOAT(MDIF) / NRUN
      PRINT *,X(I),PB(I),PA(I),PDIF(I)
      PRINT (7,*) X(I),",",PB(I),",",PA(I),",",PDIF(I)
      IF(I .LT. NP) GO TO 1
      END
      SUBROUTINE IGTAB(GI)
      DIMENSION GI(1001)
C-SETSUP TABLE OF INTEGRAL OF GAUSSIAN, FROM XMIN TO X OF EXP(-X*X/2)
      PRINT *,"DOING INTEGRAL GAUSSIAN TABLE."
      XMIN = -3.0 $ XMAX = 3.0
      NP = 1001 $ NDIV = 10
      DX = (XMAX-XMIN) / FLOAT(NP-1)
      CONST = 1./SQRT(8.*ATAN(1.))
```

```
      XOLD = XMIN $ GI(1) = 0.0
      I = 0
    1 I = I+1
      X = XMIN + DX*FLOAT(I-1)
      IF ( I .EQ. 1 ) GO TO 2
      CALL SIMP(XOLD,X,NDIV,ANS)
      GI(I) = GI(I-1) + CONST*ANS
      XOLD = X
    2 IF ( I .LT. NP ) GO TO 1
      PRINT *,"TABLE DONE, PROCEEDING TO SIMULATION. "
      RETURN
      END
      SUBROUTINE SIMP(XMIN,XMAX,N,ANS)
C-INTEGRATES A FUNCTION BY SIMPSON'S RU.E
      H = (XMAX - XMIN ) / FLOAT(N)
      NODD = N-1
      NEVEN = N-2
      ODSUM = EVSUM = 0.0
      IODD = -1
      IEVEN = 0
    1 IODD = IODD + 2
      X = XMIN + IODD*H
      Y = F1(X)
      ODSUM = ODSUM + Y
      IF ( IODD .LT. NODD ) GO TO 1
    2 IEVEN = IEVEN + 2
      X = XMIN + IEVEN * H
      Y = F1(X)
      EVSUM = EVSUM + Y
      IF ( IEVEN .LT. NEVEN ) GO TO 2
      YMIN = F1(XMIN)
      YMAX = F1(XMAX)
      ANS = H/3.0 * (YMIN + 4.0*ODSUM + 2.0*EVSUM + YMAX)
      RETURN
      END
      FUNCTION F1(X)
      F1 = EXP(-X*X/2.0)
      RETURN
      END
      SUBROUTINE STORIT(GI,ED,NE,NVAR)
      DIMENSION GI(1001),X(101),ED(21,101)
C-THIS INTERPOLATES THE SET OF EDGE RAMPS TO BE USED LATER.
      IF(NVAR .LE. 10) GO TO 1
      PRINT *,"NVAR, ",NVAR,", SHOULD BE 10 OR LESS."
      STOP
    1 IBASE = NE-2*NVAR-1
      NRAMPS = 4*NVAR+1
      DO 2 IRAMP = 1,NRAMPS
      NWIDE = IBASE + IRAMP
      CALL LINTERP(GI,X,1001,NWIDE)
      DO 2 J = 1,NWIDE
```

```
    2 ED(IRAMP,J) = X(J)
      RETURN
      END
      SUBROUTINE LINTERP(DATA1,DATA2,NP,NC)
      DIMENSION DATA1(1001),DATA2(101)
C-PICKS OR LINEARLY INTERPS. BIG DATA SET TO GET SMALLER ONE.
      SKIP = FLOAT(NP-1) / FLOAT(NC-1)
      IFLAG = MOD( (NP-1), (NC-1) )
      IF ( IFLAG .NE. 0 ) GO TO 1
C-INTERP BY PICKING OFF POINTS, E.G. EVERY 10TH OF 1000.
      IC = 0
    2 IC = IC + 1
      JC = SKIP*FLOAT(IC-1)+1
      DATA2(IC) = DATA1(JC)
      IF ( IC .LT. NC) GO TO 2
      GO TO 3
C-INTERP. BY PICKING LINEARLY BETWEEN POINTS.
    1 IC = 0
    4 IC = IC+1
      XVAL = SKIP*FLOAT(IC-1) +1
      I1 = XVAL $ Q = XVAL-I1
      DATA2(IC) = DATA1(I1) + Q*(DATA1(I1+1)-DATA1(I1))
      IF ( IC .LT. NC) GO TO 4
    3 RETURN
      END
      SUBROUTINE SETEJ(NLEFT,NRIGHT,EJ)
      DIMENSION EJ(101)
C-MAKES INTERPED. GAUSIAN RAMP RISE FROM 0 TO 1 BETWEEN NLEFT & NRIGHT.
      NWIDE=NRIGHT-NLEFT+1$NWIDE1=NWIDE+1$ISHIFT=NLEFT-1$IPAST=NRIGHT+1
C-TRIAL WIDTH, 1 MORE, NO.POINTS TO SHIFT TO CENTER, 1 PAST RIGHT END.
      DO 1 I = IPAST,101
    1 EJ(I) = 1.0
      DO 2 I = 1,NWIDE
      J = NWIDE1-I $ K = J+ISHIFT
    2 EJ(K) = EJ(J)
      DO 3 I = 1,ISHIFT
    3 EJ(I) = 0.0
      RETURN
      END
      SUBROUTINE FILLEJ(VSIG,VNOI,EJ)
      DIMENSION EJ(101)
C-SCALES EDGE TO MAX OF GRAMAX WITH SIGNAL & NOISE CONTRIBUTIONS.
      DO 1 I = 1,101
      CALL BOXNO(A,B)
C-GAUSSIAN RANDOM NOISE GENER. GIVES 2 USABLE VALUES.
C-NOW SCALE SIGNAL BY VSIG AND ADD NOISE TIMES VNOI.
    1 EJ(I) = VSIG*EJ(I) + VNOI*A
      RETURN
      END
      SUBROUTINE MD1DW(NW,EJ)
      DIMENSION EJ(101),SORT(19)
```

```
C-ONE-D MEDIAN FILTERS EJ.
      MDN1 = NW/2 $ MDN = MDN1+1
      NDO = 101-NW+1
C-NO. OF APPLICATIONS OF WINDOW.
      DO 1 I = 1,NDO
      DO 2 J = 1,NW
      K = I + (J-1)
    2 SORT(J) = EJ(K)
      CALL DSORT(SORT,NW)
      L = I + MDN1
    1 EJ(L) = SORT(MDN)
      RETURN
      END
      SUBROUTINE DSORT(X,N)
      DIMENSION X(19)
C-DOES A DROP SORT OF UP TO 19 NUMBERS.
      I = 1
    1 IF(X(I) .GT. X(I+1) ) GO TO 2
    3 I = I+1
      IF ( I .LT. N) GO TO 1
      GO TO 4
    2 CALL SWAP( X(I), X(I+1) )
      J = I-1
    5 IF(J .EQ. 0 ) GO TO 3
      IF( X(J) .GT. X(J+1) ) CALL SWAP(X(J), X(J+1) )
      J = J-1
      GO TO 5
    4 RETURN
      END
      SUBROUTINE SWAP(X,Y)
      Z = X $ X = Y $ Y = Z
      RETURN
      END
      SUBROUTINE FINDEJ(NE,NL1,NR1,NVAR,ED,EJ,VSIG,GRAMAX,ERVAL,NL2,NR2)
      DIMENSION ED(21,101),EJ(101),EJSIG(101)
C-FINDS BEST EDGE FIT TO FILTERED DATA, (ONE WITH LOWEST MEAN SQUARE
C-ERROR), AND GIVES LEFT AND RIGHT HAND END COORDINATES.
      IF(NE .LT. (2*NVAR+2) ) NVAR = (NE-2)/2
      IF(NE .GT. (101-2*NVAR) ) NVAR = (101-NE)/2
      IBASE = NE-2*NVAR-1
      ERVAL = 4.0*101.0*GRAMAX**2
      IPASS = 0
    1 IPASS = IPASS + 1
      IF(IPASS .EQ. 2) GO TO 2
      IMIN=1 $ IMAX=2*NVAR+1 $ NTIME=0 $ SIGN=1.0
      GO TO 3
    2 IMIN=2*NVAR+2 $ IMAX=4*NVAR+1 $ NTIME=2*NVAR+1 $ SIGN=-1.0
    3 DO 4 IRAMP = IMIN,IMAX
      NTIME = NTIME+SIGN*1.0
      NWIDE = IBASE + IRAMP
      IF(IPASS .EQ. 1) NLWORK = NL1 + NVAR
```

```
      IF(IPASS.EQ.1) NRWORK = NLWORK + NWIDE -1
      IF(IPASS .EQ. 2) NRWORK = NR1+NVAR
      IF(IPASS .EQ. 2) NLWORK = NRWORK-NWIDE+1
      ITIME = 0
    5 ITIME = ITIME + 1
      DO 6 J = 1,NWIDE
    6 EJSIG(J) = ED(IRAMP,J)
      CALL SETEJ(NLWORK,NRWORK,EJSIG)
      DO 7 J = NLWORK,101
    7 EJSIG(J) = EJSIG(J) * VSIG
      CALL MSE(EJSIG,EJ,ERROR)
C-FINDS MEANSQUARE ERROR OR NOISE OF EDGE OVER PURESIGNAL EJSIG.
      IF(ERROR.GE. ERVAL) GO TO 8
      ERVAL = ERROR
      NL2 = NLWORK $ NR2 = NRWORK
    8 NLWORK=NLWORK-1 $ NRWORK=NRWORK-1
    4 IF(ITIME .LT. NTIME) GO TO 5
      IF(IPASS .LT. 2) GO TO 1
      RETURN
      END
      SUBROUTINE MSE(X,Y,ERROR)
      DIMENSION X(101),Y(101)
      ERROR = 0.0
C-THIS CALCULATES THE MEAN SQUARE ERROR OF Y FROM X.
      DO 1 I = 1,101
    1 ERROR = ERROR + ( Y(I)-X(I) )**2
      RETURN
      END
```

APPENDIX C

PROGRAM EJSHOW

```
      PROGRAM EJSHOW(INPUT,OUTPUT,TAPE61=100,TAPE62=100)
      DIMENSION GI(1001),ED(21,101),X(101),SG(101),EU(101),SUF(101),
     &SUP(101),ES(101),SSF(101),SSP(101)
      CALL CONNEC(5LINPUT) $ CALL CONNEC(6LOUTPUT)
      NP = 101 $ NVAR = 5
      DO 1 I = 1,NP
    1 X(I) = FLOAT(I-1)
      XMIN = X(1) $ XMAX = X(101)
      DX = (XMAX-XMIN)/FLOAT(NP-1)
      FUZZ = .0001*DX
      GRAMAX = 255. $ SIGMAS = 3.
      CALL IGTAB(GI)
      PRINT *,"ENTER LEFT & RIGHT RAMP ENDS & % NOISE. "
      READ *,NLG,NRG,PN
      PRINT *,"ENTER WINDOW WIDTH. "
      READ *,NW
      PRINT *,"ENTER LEFT, RIGHT, UPPER & LOWER MARGIN PORTIONS. "
      READ *,XLM,RM,TM,BM
      NWG = NRG-NLG+1 $ IBASE = NWG-2*NVAR-1
      CALL STORIT(GI,ED,NWG,NVAR)
      VNOI = PN/100.*GRAMAX/SIGMAS
      VSIG = (1.-PN/100.)*GRAMAX
      IRAMP = NWG-IBASE
      DO 2 J = 1,NWG
    2 SG(J) = ED(IRAMP,J)
      CALL SETEJ(NLG,NRG,SG)
      DO 3 J = NLG,NP
    3 SG(J) = SG(J) *VSIG
    4 IRAMP = NWG-IBASE
      DO 5 J = 1,NP
      CALL BOXNO(A,B)
    5 EU(J)=SG(J)+VNOI*A
      CALL MINMAX(EU,NP,YMIN,YMAX)
      YMAX=YMAX+FUZZ $ YMIN=YMIN-FUZZ
      CALL MARGIN(XMIN,XMAX,YMIN,YMAX,XLM,RM,TM,BM)
      CALL FINDEJ(NWG,NLG,NRG,NVAR,ED,EU,VSIG,GRAMAX,ERUF,NLUF,NRUF)
      CALL MSE(SG,EU,ERUG)
      ERUF=SQRT(ERUF/101.) $ ERUG=SQRT(ERUG/101.)
      ERUF=100.*ERUF*SIGMAS/GRAMAX $ ERUG=100.*ERUG*SIGMAS/GRAMAX
      NWUF=NRUF-NLUF+1 $ IRAMP=NWUF-IBASE
      DO 6 J = 1,NWUF
    6 SUF(J) = ED(IRAMP,J)
      CALL SETEJ(NLUF,NRUF,SUF)
      DO 7 J = NLUF,NP
    7 SUF(J) = SUF(J)*VSIG
      DO 8 J = 1,NP
    8 ES(J) = EU(J)
      CALL MD1DW(NP,NW,ES)
```

```
      CALL FINDEJ(NWG,NLG,NRG,NVAR,ED,ES,VSIG,GRAMAX,ERSF,NLSF,NRSF)
      CALL MSE(SG,ES,ERSG)
      ERSF=SQRT(ERSF/101.) $ ERSG=SQRT(ERSG/101.)
      ERSF=100.*ERSF*SIGMAS/GRAMAX $ ERSG=100.*ERSG*SIGMAS/GRAMAX
      NWSF=NRSF-NLSF+1 $ IRAMP=NWSF-IBASE
      DO 9 J = 1,NWSF
    9 SSF(J) = ED(IRAMP,J)
      CALL SETEJ(NLSF,NRSF,SSF)
      DO 10 J = NLSF,NP
   10 SSF(J) = SSF(J) * VSIG
      PRINT 100
  100 FORMAT(1H ,10X,"GIVEN EDGE",2X,"GIVEN ERROR",1X,"FOUND EDGE",2X,
     &"FOUND ERROR")
      PRINT 200,NLG,NRG,ERUG,NLUF,NRUF,ERUF
  200 FORMAT(1H ,"UNSMOOTH",2X,2(I2,1X,I2,7X,F10.7,2X))
      PRINT 300,NLG,NRG,ERSG,NLSF,NRSF,ERSF
  300 FORMAT(1H ,"SMOOTHED",2X,2(I2,1X,I2,7X,F10.7,2X))
      PRINT *,"PLOT THESE RESULTS? "
      READ 400,IANS
  400 FORMAT(A3)
      IF(IANS.EQ."NO") GO TO 4
      PRINT *,"ENTER ENDS OF THIRD UNSMOOTHED SIGNAL. "
      READ *,NLUP,NRUP
      PRINT *,"ENTER ENDS OF THIRD SMOOTHED SIGNAL. "
      READ *,NLSP,NRSP
      NWUP=NRUP-NLUP+1 $ NWSP=NRSP-NLSP+1
      IRAMP = NWUP - IBASE
      DO 11 J = 1,NWUP
   11 SUP(J) = ED(IRAMP,J)
      CALL SETEJ(NLUP,NRUP,SUP)
      DO 12 J = NLUP,NP
   12 SUP(J) = SUP(J) * VSIG
      IRAMP = NWSP-IBASE
      DO 13 J = 1,NWSP
   13 SSP(J) = ED(IRAMP,J)
      CALL SETEJ(NLSP,NRSP,SSP)
      DO 14 J = NLSP,NP
   14 SSP(J) = SSP(J)*VSIG
      PRINT *,"CHOSEN RAMPS SETUP."
      CALL PLOT(X,EU,SUF,SUP,NP,XMIN,XMAX,YMIN,YMAX)
      CALL PLOT(X,ES,SSF,SSP,NP,XMIN,XMAX,YMIN,YMAX)
   15 PRINT *,"ENTER WINDOW WIDTH. "
      READ *,NW
      DO 16 J = 1,NP
   16 ES(J) = EU(J)
      CALL MDlDW(NP,NW,ES)
      CALL FINDEJ(NWG,NLG,NRG,NVAR,ED,ES,VSIG,GRAMAX,ERSF,NLSF,NRSF)
      CALL MSE(SG,ES,ERSG)
      ERSF=SQRT(ERSF/101.) $ ERSG=SQRT(ERSG/101.)
      ERSF=100.*ERSF*SIGMAS/GRAMAX $ ERSG=100.*ERSG*SIGMAS/GRAMAX
      NWSF=NRSF-NLSF+1 $ IRAMP=NWSF-IBASE
```

```
      DO 17 J = 1,NWSF
   17 SSF(J) = ED(IRAMP,J)
      CALL SETEJ(NLSF,NRSF,SSF)
      DO 18 J = NLSF,NP
   18 SSF(J) = SSF(J) *VSIG
      PRINT 100
      PRINT 300,NLG,NRG,ERSG,NLSF,NRSF,ERSF
      PRINT *,"ENTER ENDS OF THIRD SMOOTHED SIGNAL. "
      READ *,NLSP,NRSP
      NWSP=NRSP-NLSP+1 $ IRAMP=NWSP-IBASE
      DO 19 J = 1,NWSP
   19 SSP(J) = ED(IRAMP,J)
      CALL SETEJ(NLSP,NRSP,SSP)
      DO 20 J = NLSP,NP
   20 SSP(J) = SSP(J) * VSIG
      PRINT *,"CHOSEN RAMP SETUP."
      CALL PLOT(X,ES,SSF,SSP,NP,XMIN,XMAX,YMIN,YMAX)
      GO TO 15
      END
      SUBROUTINE PLOT(X,Y1,Y2,Y3,NP,XMIN,XMAX,YMIN,YMAX)
      DIMENSION X(101),Y1(101),Y2(101),Y3(101)
      READ 100,ICHAR
  100 FORMAT(A4)
      CALL INITT(30) $ CALL TERM(2,4096)
      CALL DWINDO(XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA( X(1), Y1(1) )
      DO 1 I = 2,NP
    1 CALL DRAWA(X(I), Y1(I) )
      CALL VCURSR(IC,IX,IY)
      CALL MOVEA(X(1), Y2(1))
      DO 2 I = 2,NP
    2 CALL DASHA(X(I), Y2(I),12)
      CALL VCURSR(IC,IX,IY)
      CALL MOVEA(X(1),Y3(1))
      DO 3 I = 2,NP
    3 CALL DASHA(X(I),Y3(I),34)
      CALL ANMODE $ READ 100,ICHAR
      IF(ICHAR .EQ. "STOP" ) STOP
      RETURN
      END
      SUBROUTINE IGTAB(GI)
      DIMENSION GI(1001)
C-SETSUP TABLE OF INTEGRAL OF GAUSSIAN, FROM XMIN TO X OF EXP(-X*X/2)
      PRINT *,"DOING INTEGRAL GAUSSIAN TABLE."
      XMIN = -3.0 $ XMAX = 3.0
      NP = 1001 $ NDIV = 10
      DX = (XMAX-XMIN) / FLOAT(NP-1)
      CONST = 1./SQRT(8.*ATAN(1.))
      XOLD = XMIN $ GI(1) = 0.0
      I = 0
    1 I = I+1
```

```
      X = XMIN + DX*FLOAT(I-1)
      IF ( I .EQ. 1 ) GO TO 2
      CALL SIMP(XOLD,X,NDIV,ANS)
      GI(I) = GI(I-1) + CONST*ANS
      XOLD = X
    2 IF ( I .LT. NP ) GO TO 1
      PRINT *,"TABLE DONE, PROCEEDING TO SIMULATION. "
      RETURN
      END
      SUBROUTINE SIMP(XMIN,XMAX,N,ANS)
C-INTEGRATES A FUNCTION BY SIMPSON'S RU.E
      H = (XMAX - XMIN ) / FLOAT(N)
      NODD = N-1
      NEVEN = N-2
      ODSUM = EVSUM = 0.0
      IODD = -1
      IEVEN = 0
    1 IODD = IODD + 2
      X = XMIN + IODD*H
      Y = F1(X)
      ODSUM = ODSUM + Y
      IF ( IODD .LT. NODD ) GO TO 1
    2 IEVEN = IEVEN + 2
      X = XMIN + IEVEN * H
      Y = F1(X)
      EVSUM = EVSUM + Y
      IF ( IEVEN .LT. NEVEN ) GO TO 2
      YMIN = F1(XMIN)
      YMAX = F1(XMAX)
      ANS = H/3.0 * (YMIN + 4.0*ODSUM + 2.0*EVSUM + YMAX)
      RETURN
      END
      FUNCTION F1(X)
      F1 = EXP(-X*X/2.0)
      RETURN
      END
      SUBROUTINE LINTERP(DATA1,DATA2,NP,NC)
      DIMENSION DATA1(1001),DATA2(101)
C-PICKS OR LINEARLY INTERPS. BIG DATA SET TO GET SMALLER ONE.
      SKIP = FLOAT(NP-1) / FLOAT(NC-1)
      IFLAG = MOD( (NP-1), (NC-1) )
      IF ( IFLAG .NE. 0 ) GO TO 1
C-INTERP BY PICKING OFF POINTS, E.G. EVERY 10TH OF 1000.
      IC = 0
    2 IC = IC + 1
      JC = SKIP*FLOAT(IC-1)+1
      DATA2(IC) = DATA1(JC)
      IF ( IC .LT. NC) GO TO 2
      GO TO 3
C-INTERP. BY PICKING LINEARLY BETWEEN POINTS.
    1 IC = 0
```

```
    4 IC = IC+1
      XVAL = SKIP*FLOAT(IC-1) +1
      I1 = XVAL $ Q = XVAL-I1
      DATA2(IC) = DATA1(I1) + Q*(DATA1(I1+1)-DATA1(I1))
      IF ( IC .LT. NC) GO TO 4
    3 RETURN
      END
      SUBROUTINE SETEJ(NLEFT,NRIGHT,EJ)
      DIMENSION EJ(101)
C-MAKES INTERPED. GAUSIAN RAMP RISE FROM 0 TO 1 BETWEEN NLEFT & NRIGHT.
      NWIDE=NRIGHT-NLEFT+1$NWIDE1=NWIDE+1$ISHIFT=NLEFT-1$IPAST=NRIGHT+1
C-TRIAL WIDTH, 1 MORE, NO.POINTS TO SHIFT TO CENTER, 1 PAST RIGHT END.
      DO 1 I = IPAST,101
    1 EJ(I) = 1.0
      DO 2 I = 1,NWIDE
      J = NWIDE1-I $ K = J+ISHIFT
    2 EJ(K) = EJ(J)
      DO 3 I = 1,ISHIFT
    3 EJ(I) = 0.0
      RETURN
      END
      SUBROUTINE FILLEJ(VSIG,VNOI,EJ)
      DIMENSION EJ(101)
C-SCALES EDGE TO MAX OF GRAMAX WITH SIGNAL & NOISE CONTRIBUTIONS.
      DO 1 I = 1,101
      CALL BOXNO(A,B)
C-GAUSSIAN RANDOM NOISE GENER. GIVES 2 USABLE VALUES.
C-NOW SCALE SIGNAL BY VSIG AND ADD NOISE TIMES VNOI.
    1 EJ(I) = VSIG*EJ(I) + VNOI*A
      RETURN
      END
      SUBROUTINE MD1DW(NP,NW,EJ)
      DIMENSION EJ(1001),SORT(19)
C-ONE-D MEDIAN FILTERS EJ.
      MDN1 = NW/2 $ MDN = MDN1+1
      NDO = NP-NW+1
C-NO. OF APPLICATIONS OF WINDOW.
      DO 1 I = 1,NDO
      DO 2 J = 1,NW
      K = I + (J-1)
    2 SORT(J) = EJ(K)
      CALL DSORT(SORT,NW)
      L = I + MDN1
    1 EJ(L) = SORT(MDN)
      RETURN
      END
      SUBROUTINE DSORT(X,N)
      DIMENSION X(19)
C-DOES A DROP SORT OF UP TO 19 NUMBERS.
      I = 1
    1 IF(X(I) .GT. X(I+1) ) GO TO 2
```

```
    3 I = I+1
      IF ( I .LT. N) GO TO 1
      GO TO 4
    2 CALL SWAP( X(I), X(I+1) )
      J = I-1
    5 IF(J .EQ. 0 ) GO TO 3
      IF( X(J) .GT. X(J+1) ) CALL SWAP(X(J), X(J+1) )
      J = J-1
      GO TO 5
    4 RETURN
      END
      SUBROUTINE SWAP(X,Y)
      Z = X $ X = Y $ Y = Z
      RETURN
      END
      SUBROUTINE MSE(X,Y,ERROR)
      DIMENSION X(101),Y(101)
      ERROR = 0.0
C-THIS CALCULATES THE MEAN SQUARE ERROR OF Y FROM X.
      DO 1 I = 1,101
    1 ERROR = ERROR + ( Y(I)-X(I) )**2
      RETURN
      END
      SUBROUTINE MINMAX(Y,NP,YMIN,YMAX)
      DIMENSION Y(1001)
      YMIN = YMAX = Y(1)
      DO 1 I = 1,NP
      IF ( Y(I) .LT. YMIN ) YMIN = Y(I)
    1 IF ( Y(I) .GT. YMAX ) YMAX = Y(I)
      RETURN
      END
      SUBROUTINE MARGIN(XMIN,XMAX,YMIN,YMAX, XLM,RM,TM,BM)
      HH = (XMAX-XMIN) / (1.0-XLM-RM)
      HV = (YMAX-YMIN) / (1.0-TM-BM)
      XMIN = XMIN -HH*XLM $ XMAX = XMAX + HH*RM
      YMAX = YMAX + HV*TM $ YMIN = YMIN - HV*BM
      RETURN
      END
      SUBROUTINE STORIT(GI,ED,NE,NVAR)
      DIMENSION GI(1001),X(101),ED(21,101)
C-THIS INTERPOLATES THE SET OF EDGE RAMPS TO BE USED LATER.
      IF(NVAR .LE. 10) GO TO 1
      PRINT *,"NVAR, ",NVAR,", SHOULD BE 10 OR LESS."
      STOP
    1 IBASE = NE-2*NVAR-1
      NRAMPS = 4*NVAR+1
      DO 2 IRAMP = 1,NRAMPS
      NWIDE = IBASE + IRAMP
      CALL LINTERP(GI,X,1001,NWIDE)
      DO 2 J = 1,NWIDE
    2 ED(IRAMP,J) = X(J)
```

```
      RETURN
      END
      SUBROUTINE FINDEJ(NE,NL1,NR1,NVAR,ED,EJ,VSIG,GRAMAX,ERVAL,NL2,NR2)
      DIMENSION ED(21,101),EJ(101),EJSIG(101)
C-FINDS BEST EDGE FIT TO FILTERED DATA, (ONE WITH LOWEST MEAN SQUARE
C-ERROR), AND GIVES LEFT AND RIGHT HAND END COORDINATES.
      IF(NE .LT. (2*NVAR+2) ) NVAR = (NE-2)/2
      IF(NE .GT. (101-2*NVAR) ) NVAR = (101-NE)/2
      IBASE = NE-2*NVAR-1
      ERVAL = 4.0*101.0*GRAMAX**2
      IPASS = 0
    1 IPASS = IPASS + 1
      IF(IPASS .EQ. 2) GO TO 2
      IMIN=1 $ IMAX=2*NVAR+1 $ NTIME=0 $ SIGN=1.0
      GO TO 3
    2 IMIN=2*NVAR+2 $ IMAX=4*NVAR+1 $ NTIME=2*NVAR+1 $ SIGN=-1.0
    3 DO 4 IRAMP = IMIN,IMAX
      NTIME = NTIME+SIGN*1.0
      NWIDE = IBASE + IRAMP
      IF(IPASS .EQ. 1) NLWORK = NL1 + NVAR
      IF(IPASS.EQ.1) NRWORK = NLWORK + NWIDE -1
      IF(IPASS .EQ. 2) NRWORK = NR1+NVAR
      IF(IPASS .EQ. 2) NLWORK = NRWORK-NWIDE+1
      ITIME = 0
    5 ITIME = ITIME + 1
      DO 6 J = 1,NWIDE
    6 EJSIG(J) = ED(IRAMP,J)
      CALL SETEJ(NLWORK,NRWORK,EJSIG)
      DO 7 J = NLWORK,101
    7 EJSIG(J) = EJSIG(J) * VSIG
      CALL MSE(EJSIG,EJ,ERROR)
C-FINDS MEANSQUARE ERROR OR NOISE OF EDGE OVER PURESIGNAL EJSIG.
      IF(ERROR.GE. ERVAL) GO TO 8
      ERVAL = ERROR
      NL2 = NLWORK $ NR2 = NRWORK
    8 NLWORK=NLWORK-1 $ NRWORK=NRWORK-1
    4 IF(ITIME .LT. NTIME) GO TO 5
      IF(IPASS .LT. 2) GO TO 1
      RETURN
      END
```

APPENDIX D

PROGRAM PULPLT

```
      PROGRAM PULPLT(INPUT,OUTPUT,TAPE61=100,TAPE62=100)
      DIMENSION X(1001),Y(1001),YF(1001)
      CALL CONNEC(5LINPUT) $ CALL CONNEC(6LOUTPUT)
      GRAMAX=255. $ SIGMAS=3.
      PRINT *,"ENTER % NOISE & % PULSEWIDTH "
      READ *,PN,PPW
      PRINT *,"ENTER MIN&MAX DEG.S & NO.POINTS "
      READ *,XMIN,XMAX,NP
      PRINT *,"ENTER LEFT, RIGHT, UPPER & LOWER MARGIN PORTIONS "
      READ *,XLM,RM,TM,BM
      VNOI = PN/100.*GRAMAX/SIGMAS
      VSIG = (1.-PN/100.)*GRAMAX
      PWIDTH=PPW/100.*(XMAX-XMIN)
      XMID=(XMIN+XMAX)/2.0
      PMIN = XMID-PWIDTH/2.0 $ PMAX = XMID+PWIDTH/2.0
      DX = (XMAX-XMIN)/FLOAT(NP-1)
      I = 0
    1 I = I+1
      X(I) = XMIN+DX*FLOAT(I-1)
      AMP = 1.0
      IF(X(I) .LT. PMIN .OR. X(I) .GT. PMAX) AMP = 0.0
      CALL BOXNO(A,B)
      Y(I) = VSIG*AMP + VNOI*A
      IF(I .LT. NP) GO TO 1
      CALL MINMAX(Y,NP,YMIN,YMAX)
      FUZZ = .0001*DX
      YMAX = YMAX+FUZZ $ YMIN=YMIN-FUZZ
      CALL MARGIN(XMIN,XMAX,YMIN,YMAX,XLM,RM,TM,BM)
      CALL PLOT(X,Y,NP,XMIN,XMAX,YMIN,YMAX)
    2 PRINT *,"ENTER WINDOW WIDTH. "
      READ *,NW
      DO 3 I = 1,NP
    3 YF(I) = Y(I)
      CALL MD1DW(NP,NW,YF)
      CALL PLOT(X,YF,NP,XMIN,XMAX,YMIN,YMAX)
      GO TO 2
      END
      SUBROUTINE MD1DW(NP,NW,EJ)
      DIMENSION EJ(1001),SORT(19)
C-ONE-D MEDIAN FILTERS EJ.
      MDN1 = NW/2 $ MDN = MDN1+1
      NDO = NP-NW+1
C-NO. OF APPLICATIONS OF WINDOW.
      DO 1 I = 1,NDO
      DO 2 J = 1,NW
      K = I + (J-1)
    2 SORT(J) = EJ(K)
      CALL DSORT(SORT,NW)
```

```
      L = I + MDN1
    1 EJ(L) = SORT(MDN)
      RETURN
      END
      SUBROUTINE DSORT(X,N)
      DIMENSION X(19)
C-DOES A DROP SORT OF UP TO 19 NUMBERS.
      I = 1
    1 IF(X(I) .GT. X(I+1) ) GO TO 2
    3 I = I+1
      IF ( I .LT. N) GO TO 1
      GO TO 4
    2 CALL SWAP( X(I), X(I+1) )
      J = I-1
    5 IF(J .EQ. 0 ) GO TO 3
      IF( X(J) .GT. X(J+1) ) CALL SWAP(X(J), X(J+1) )
      J = J-1
      GO TO 5
    4 RETURN
      END
      SUBROUTINE SWAP(X,Y)
      Z = X $ X = Y $ Y = Z
      RETURN
      END
      SUBROUTINE MINMAX(Y,NP,YMIN,YMAX)
      DIMENSION Y(1001)
      YMIN = YMAX = Y(1)
      DO 1 I = 1,NP
      IF ( Y(I) .LT. YMIN ) YMIN = Y(I)
    1 IF ( Y(I) .GT. YMAX ) YMAX = Y(I)
      RETURN
      END
      SUBROUTINE MARGIN(XMIN,XMAX,YMIN,YMAX, XLM,RM,TM,BM)
      HH = (XMAX-XMIN) / (1.0-XLM-RM)
      HV = (YMAX-YMIN) / (1.0-TM-BM)
      XMIN = XMIN -HH*XLM $ XMAX = XMAX + HH*RM
      YMAX = YMAX + HV*TM $ YMIN = YMIN - HV*BM
      RETURN
      END
      SUBROUTINE PLOT(X,Y,NP, XMIN,XMAX,YMIN,YMAX)
      DIMENSION X(1001),Y(1001)
  100 FORMAT(A4)
      CALL INITT(30) $ CALL TERM(2,4096)
      CALL DWINDO(XMIN,XMAX,YMIN,YMAX)
      CALL MOVEA( X(1), Y(1) )
      DO 1 I = 2,NP
    1 CALL DRAWA( X(I), Y(I) )
      CALL ANMODE $ READ 100,ICHAR
      IF(ICHAR.EQ."STOP") STOP
      RETURN
      END
```

DISTRIBUTION LIST

Director
U.S. Army Research Office
ATTN: Library
P.O. Box 12211
Research Triangle Park, NC 27709

Director
Department of Defense
Advanced Research and Projects Agency
Washington, DC 20301

Commander
U.S. Army Aviation Systems Command
12th and Spruce Streets
St. Louis, MO 63166

Commander
U.S. Army Tank-Automotive Research and
Development Command
ATTN: DRDTA, Technical Library
Warren, MI 48090

Commander
U.S. Army Electronics Command and
Devices Laboratory
Ft Monmouth, NJ 07703

Commander
U.S. Army Electronics Command
Atmospheric Science Laboratory
White Sands Missile Range
ATTN: DRSEL-BL-MS, R. Gomez
ATAA-SL
White Sands, NM 88002

Commander
U.S. Army TECOM
ATTN: AMSTE-TA-A (2)
STEAP-TL
STEAP-DS-LP
Aberdeen Proving Ground, MD 21005

Commander
U.S. Army Harry Diamond Laboratories
ATTN: AMXDO-TD/002
AMXDO-TIB
2800 Powder Mill Road
Adelphi, MD 20783

Commander
U.S. Army Armament Research and
Development Command
ATTN: DRDAR-SC (2)
DRDAR-SCF
DRDAR-SCF-I
DRDAR-SCF-IO
DRDAR-SCP
DRDAR-MSA
DRDAR-TSS (5)
DRDAR-GCL
Dover, NJ 07801

Administration
Defense Technical Information Center
ATTN: Accessions Division (12)
Cameron Station
Alexandria, VA 22314

Commander
U.S. Army Armament Materiel and
Readiness Command
ATTN: DRDAR-LEP-L
Rock Island, IL 61299

Director
U.S. Army Materiel Systems
Analysis Activity
ATTN: DRXSY-MP
Aberdeen Proving Ground, MD 21005

Commander/Director
Chemical Systems Laboratory
U.S. Army Armament Research and
Development Command
ATTN: DRDAR-CLJ-L
DRDAR-CLB-PA
APG, Edgewood Area, MD 21010

Chief
Benet Weapons Laboratory, LCWSL
U.S. Army Armament Research and
Development Command
ATTN: DRDAR-LCB-TL
Watervliet, NY 12189

Thomas S. Huang
Institute for Image Technology
1000 North Western Avenue
West Lafayette, IN 47906

Gary Sivak (10)
5862 Troy Villa Boulevard
Huber Heights, OH 45424